Verification of Security Properties Using Formal Techniques

Dissertation zur Erlangung des Doktorgrades
der Mathematisch und Informatik Fakultäten
der Georg-August-Universität zu Göttingen

vorgelegt von
Saleh Al-shadly
aus Hadhramout (der Jemen)

Göttingen 2013

Mitglieder der Prüfungskommission:

Referent: **Prof. Dr. Dieter Hogrefe**
Institut für Informatik, Georg-August-Universität Göttingen

Korreferent: **Prof. Dr. Xiaoming Fu**
Institut für Informatik, Georg-August-Universität Göttingen

Weitere Mitglieder der Prüfungskommission:

Prof. Dr. Jens Grabowski
Institut für Informatik, Georg-August-Universität Göttingen

Prof. Dr. Eckart Modrow
Institut für Informatik, Georg-August-Universität Göttingen

Prof. Dr. Konrad Rieck
Institut für Informatik, Georg-August-Universität Göttingen

Prof. Dr. Stephan Waack
Institut für Informatik, Georg-August-Universität Göttingen

Tag der mündlichen Prüfung: April 9^{th} 2013

Abstract

Nowadays, the necessity of developing collaborative and distributed computing systems makes networks, and specially Internet, the key element of the system design process. Several new applications have been appeared in the Internet which require a variety of security requirements must be fulfilled. There are applications that exchange sensitive and private information which must be secret and authentic. Other applications (i.e. real-time applications) which require a certain level of quality of service should be maintained. Thus, the traditional solutions can not fulfill the new requirements. Usually, requirements such as confidentiality and integrity are provided using cryptography. In current computing systems, cryptography should not only provide those requirements but also it must guarantee other security requirements such as non-repudiation, anonymity, DoS-resistance, and so on. Consequently, new security protocols must be proposed in order to cover the continuous needs.

Due to distributed nature of security protocols and the hostile environment where they are usually executed, the design of correct security protocols is difficult and error-prone. Thus, analyzing their correctness is a crucial task. Formal methods have been intensively used for analyzing confidentiality and integrity requirements and there exists several automated tools for such analysis. However, the use of formal methods for analyzing requirements such as DoS-resistance is still not mature and an emergent field. This research is concerned to study the application of formal techniques in specification and verification of security requirements focusing on authentication, secrecy and DoS-resistance requirements. Additionally, an attempt to quantify the impact of denial of service attacks on the network and evaluate their defense mechanisms is also presented.

Specific outcomes of this work include:

- A comprehensive survey of most important formal techniques and tools for specification and verification of security protocols;
- Analysis of authentication and secrecy requirements of an authentication protocol of inter-domain handover using AVISPA toolkit;
- Presenting the state-of-art of formal techniques of denial of modeling and analysis of service as well as other DoS basic knowledge such as types of DoS attacks, attack tools, defense mechanisms and strategies, etc;
- Quantifying the impact of DoS attacks on networks and evaluating their active defenses through simulation, and
- Outlook for automating Meadows's cost-based framework using probabilistic model checking.

Acknowledgments

It is not possible to complete a research degree without the considerable support of a great many people and organizations.

First of all, I would like to thank my supervisor, Prof. Dieter Hogrefe and express my deep and sincere gratitude to him for giving me the opportunity to conduct the research in his group, and for his support, guidance and invaluable advices. I also highly appriciate his kindness.

I also would like to gratitude Prof. Xiaoming Fu and Dr. Somayeh Taheri. I am greatly thankful for their valuable advises and constructive suggestions provided through my thesis.

I also would like to extend my gratitude to all the members of the examination committee of my thesis: Prof. Jenes Grabowski, Prof. Eckart Modrow, Prof. Konrad Rieck and Prof. Stephan Waack.

I thank all of the academic and administrative staff of the Institute of Computer Science at Georg-August University of Goettingen for their guidance and encouragement. In particular I would like to extend my gratitude to Carmen Scherbaum de Huamán and Annette Kadziora for making things easy in the institute. I must not forget our technical supporter Udo Burghardt, the one who having always magic tricks making computers to obey him. I would also to express my thanks to my colleagues those they have already left us Dr. Omar Alfandi and Dr. Patryk Chamuczyński.

I would also express my great thanks to few external people. I would like to thank Mrs. Andrea Gerecke from DAAD for her great help and support. Many thanks go to my colleague Wadhah Sabti from Hadhramout university for providing me a remote support of preparing my red tape papers in Mukalla. Here, I must not forget to extend my gratitude to my friends Dr. Nizar Aouni who has done a great help for providing a proof reading of my thesis. In the same matter, many thanks go to my friends Youssef Shehadeh and Salamah Alwahsh.

In addition, I would like to acknowledge the Deutscher Akademischer Austausch Dienst (DAAD) and Hadhramout university for generously providing a scholarship. I also would like to acknowledge the Institute of Computer Science at Georg-August University of Goettingen for supporting other studying expense. Without this support I would have not been able to undertake the research presented in this dissertation.

Last but not least, I would like to extend my deepest gratitude to my wife and family for their tremendous and continued support. I would not have been able to do this without them. Special thanks to my children; Fatema, Mohammed and Reem, and I hope, from now onwards, to have more time for you.

THANK YOU VERY MUCH FOR ALL AND FOR EVERYONE WHOM I KNOW AND I DID NOT MENTION HIS NAME HERE DUE TO THE LIMITED SPACE.

To
My Parents

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Contents

1.1 Overview

Secure communication is presently a very important requirement. Failures of such requirements such as unauthorized access or user impersonation have resulted in countless losses. In order to allow the exchange of sensitive information over hostile environments, such as Internet, security protocols must be present to achieve this goal. A security protocol is a communication protocol that aims to achieve security requirements, such as authenticity, integrity, key-agreement, anonymity and so on.

The fast development of new network applications (i.e., grid and cloud computing, sensor networks, financial operations and e-commerce, etc) which need the exchange of sensitive information (i.e., passwords, credit numbers, account numbers, etc). That has given great importance to the study of cryptographic protocols.

In todays communication several new security requirements (such as non-repudiation, anonymity, availability ,...) must be guaranteed. Thus, new security protocols are proposed to meet these requirements. Consequently, not only the security standards should be updated, but also the verification techniques should also be updated in order to be adapted to those new needs.

As a verification technique, *formal methods* [125] have been successfully applied to the analysis of hardware systems, software systems and communication protocols. In the 70's and 80's some initial works appeared to support the analysis of security protocols. In 90's, formal methods have been widely used for analyzing security protocols which allowed to discover some attacks on important security protocols considered to be secure for several years [50].

Formal methods are mathematical-based approaches for specifying and rigorously verifying a candidate model of the system. Specifying the system model using

notations and languages with defined mathematical meanings, allows expressing the system behavior with precision an no ambiguity. The system properties can be specified and verified related to the modeled environment. There are different specification languages which are normally defined by a syntax and a semantic. First, *process algebra* (i.e. CSP [307], CCS [264] and π-Calculus [265]) is one of a widely accepted and much used technique in the specification and verification of security protocols. Second, *graphical languages* (i.e. Timed Automata (TA) [38, 37] and Petri Nets [294, 212]).

The research on formal methods is rich and a huge body of research exist. There are several formal approaches co-existed with supporting tools supporting the analysis and verification of security protocols (and hardware/software systems in general). Those formal approaches can be classified in three groups as follows:

1. *Modal or Belief logics.* Modal logics consist of a language to describe various statements of a protocol such as what participants know or believes, and some inference rules which are used to derive new statements from the current ones. The goal of analysis is to derive a statement that presents the correctness condition of the protocol. The designer's inability to derive it indicates that the protocol may be not correct. The main advantage of these logics is their simplicity. Examples of such logics are BAN [95] and BGNY [84].

2. *Model Checking approaches.* They consider a relatively large, but finite, number of possible protocol behaviors (the state space) and allow that they satisfy a set of properties. The state space is explored using an algorithm to find out a path which leads to a state where the properties are not hold. Several model checking tools have been developed; some are general-purpose model checking tools such as Spin [203], NuSMV [118, 119], UPPAAL [68, 60], etc, and some others are devoted to security protocols specification and verification such as Casper/FDR [250], AVISPA [11] and Scyther [143], etc. Most of the latter tools follow the Dolev-Yao attacker model [162].

3. *Theorem proving methods.* They use inference rules by means of a set of theorems in oder to produce a formal proof of the protocol correctness. Theorem proving methods consider *all* possible protocol behaviors and they are well suited for proving protocol correctness, rather than finding attacks on protocols. Additionally, they have an interactive nature. An example of theorem proving is Meadows' NRL [259] as well as Paulson's inductive approach [286].

As mentioned above, the belief logics and theorem proving approaches focus on proving the correctness of the protocol based on its specifications and a set of inference rules. In contrast, model checking techniques are suitable and based on searching the state space to find incorrect traces.

However, the focus of this thesis is on using model checking as formal technique of specification, analysis and verification of security protocols. In the initial phase of the study, several security protocols have by analyzed using AVISPA [11]. AVISPA

toolbox offers four different approaches share the same input language. First, the protocol is specified using a high level language which is translated into an intermediate format that forms the common input language of the four back-ends of AVISPA toolbox.

1.2 Security Protocol Analysis as Model Checking Problem

The problem of security protocol analysis can be realized as a model checking problem similar to the general problem of software model checking [69]. In this context, the decision problem of model checking is "*given a security protocol P, a model M of the protocol and its desired security requirement Θ, determine whether the security requirement Θ* is guaranteed (holds) in $M(P, \Theta)$".

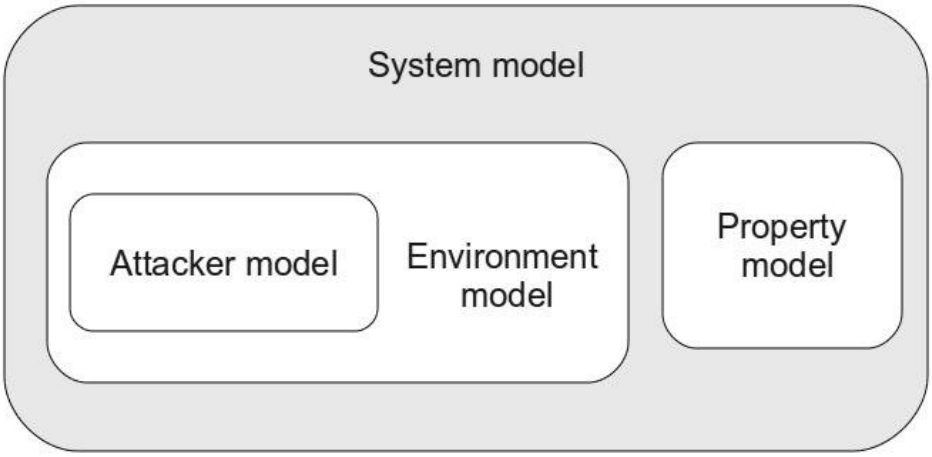
Thus, in order to analyze a security protocol using model checking techniques, as first step, we should describe an *analysis model* of the intended protocol. The analysis model is an abstract formal specifications of the protocol which reflects the properties or behaviors of the protocol. Usually, a formal language is used to describe the analysis model. As a second step, specifying *the property model* which is the formalization of the security requirements of the intended protocol (the goals which the protocol guarantees). Third, specifying *the attacker model. the attacker model* describes the capabilities and the behavior of the intruder as a participant of the protocol which does not necessary follow the protocol rules. The main objective of the attacker model is break the protocol by subverting the protocol's goal specified by the property model. Finally, description of *the environment model.* The environment model includes the honest participants of the protocol, which follow the protocol rules (or steps). Furthermore, the encoding of the security protocol and the description of the communication mechanisms between the protocol participants are also included in the environment model. Figure 1.1 depicts aforementioned models.

As described above, An analysis model of a protocol consists of three sub-models, viz., the property model, the attacker model and the environment model. The generality and flexibility of both the environment model and property model is an advantage that can allow different protocols and properties be specified. The choice of the attacker model is the *barometer* of the model precision. However, the strength of attacker model depends on the security requirements that are intended for verification. For example, the attacker model that can be use for reasoning about *safety* requirements (strong security requirements; i.e. secrecy, authenticity), may not be suitable for reasoning about *availability* requirements (i.e. DoS-resistance).

1.3 Problem At a Glance

Based on our study of different security oriented formal techniques and tools (with a special focus on model checking tools) and their application on security protocol specification and verification several problems can be pointed. Firstly, *modeling*

Figure 1.1: Security protocols analysis model



time aspects such as *timeouts* and *timestamps*; there is no devoted security model checking tool that provides a specification language that allows the use of such timed features based on global clocks in conjunction with other relevant security requirements. Using general-purpose model checking tools like UPPAAL, which supports modeling clocks, is more error prone and more difficult that using a devoted security model checking whose specification language supports security protocols. Secondly, *support of quantitative analysis*, availability requirements such as Denial-of-Service (DoS) resistance properties require a quantitative analysis. To the best of our knowledge, there is no security model that checking tool supports such kind of analysis. Furthermore, security model checking tools implement an attacker model based on the Dolev-Yao attacker model which is not suitable for analyzing the availability requirements since it has very strong assumptions. As a remedy solution to quantitatively analyze such properties, general-purpose probabilistic model checking tools can be used, however, this solution lacks of generality and it is also error prone. In this thesis, we consider using probabilistic model checking tools, for reasoning about DoS-resistance properties, in conjunction with the queuing theory fundamentals. Another solution to the same problem, we used discrete-event network simulators in order to measure the offensive severity of bandwidth consumption DoS attacks.

1.4 Thesis Objectives

This work deals with the automated specification and verification of security protocols using formal techniques. Several protocols are specified and analyzed which offer different security applications and challenges. Our focus was on authentication protocols, as an example, we specified and analyzed an inter-domain handover authentication protocol based on PANA. Our analysis shows that the protocol in most of the specified scenarios is safe except one scenario reveals a possible attack.

Another goal of this research is to provide an automated quantitative analysis of DoS resistance. Additionally, some other objectives co-exist with the main ones:

- Providing a comprehensive survey of formal methods and tools of security protocol analysis. In particular, model checking tools which are designated to automate protocol analysis.
- Studying different existing authentication protocols as well as studying different DoS-resistance solutions.
- As a result of the above studies, we pointed several open research questions and new challenges about security protocols specification and verification that the current approaches do not cover. We have attempted to provide initial solutions of the quantitative analysis of security properties by means of probabilistic model checking and queuing theory as well as network simulators.

1.5 Thesis Structure

Chapter 1: Introduction

This chapter briefly introduces the security analysis using formal techniques as well as briefly describes the thesis' work, and its objectives. The structure of the thesis is also described in this chapter.

Chapter 2: Formal Analysis of Security Protocols

In this chapter, a survey of the formal methods for the analysis of security protocols is reported. This survey gives a detailed description how can the formal methods support specification, verification, design and Synthesis of security protocols. It highlights several research works related to the area of security protocols engineering. Furthermore, we describe different kinds of attacks against security protocols and we briefly introduce security requirements (or properties) and the Dolev-Yao attacker model. Finally, we give an illustration of using model checking tools for analyzing authentication protocols.

Chapter 3: Case Study (Inter-domain Handover Protocol)

In this chapter, we analyzed the security properties (secrecy and authenticity) of an inter-domain handover protocol which is based on context transfer and PANA protocol. The analysis is done through AVISPA toolkit and we were able to identify a security vulnerability in the protocol.

Chapter 4: Denial-of-Service Attacks

This chapter, firstly, introduces the fundamental knowledge related to Denial of Service (DoS) attacks. We present Several examples of popular DoS attacks as well

as some statistics of the impact of DoS attack on the Internet. Then, the defensive techniques and resistance strategies for preventing DoS attacks in authentication protocols are described. Finally, we provide a through analysis of the state-of-art of proposed formal methods for modeling and analysis of DoS resistance.

Chapter 5: Analyzing DoS Attacks Through Simulation

This chapter focuses on analysis of DoS attack impact on the network and effectiveness of active defense mechanisms. Several QoS-based measurement metrics are used for the purpose of this analysis which is done through simulation. We simulate Distributed Denial of Service (DDoS) attacks using UDP flooding.

Chapter 6: Conclusion and Future Work

In this chapter, conclusions of the work carried out in the thesis are drawn as well as tangible contributions. An outlook of a formal method that intend to automate the cost-based framework of Meadows [261] is outlined. Finally, trends of future research, that can be accomplished from this work are also highlighted.

1.6 Acknowledgments

The work in this thesis was supported by German Academic Exchange Service (Deutscher Akademischer Austausch Dienst (DAAD))[1] and Ministry of Higher Education and Scientific Research of Yemen represented by Hadhramout university of science and technology. Without both entities' financial support, I would not have been able to conduct this research work.

[1] Website of DAAD: www.daad.de

Chapter 2

Formal Analysis of Security Protocols

Contents

2.1 Introduction

Security protocols are communication protocols aimed to provide various security services (usually called security properties or security requirements) to the participants (or entities) of the protocol. Data integrity, confidentiality, authenticity, and so on are some examples of the security services that security protocols intend to guarantee. More details are to be found Section §2.3.

Cryptographic primitives, such as one-way hash functions and encryption/decryption functions (symmetric or asymmetric), represent the basic building-blocks of security protocols. Encryption functions are used to achieve confidentiality or authentication, whereas hash functions are used to provide integrity or non-repudiation. For achieving security requirements of security protocols cryptographic primitives are used in combination than in isolation.

However, security protocols are extremely simple if only their length is considered, whereas their properties are extremely subtle. Due to this fact the informal techniques are insufficient. Several security protocols were assumed to be correct, where their weaknesses were discovered years later. The well-known Needham and Schroeder Public Key (NSPK) protocol was found to be flawed 17 years after its publication [249]. Thus, the absence of formal verification can lead to flaws and security errors within the protocols remaining undetected. Formal verification aims to provide a rigid and complete evaluation of the correctness of security protocols so that even subtle faults can be discovered.

Several methods and tools have been developed to support the formal analysis and verification of security protocols. Mainly, verification methods can be classified in three main approaches: modal (belief) logics, e.g. BAN [95] and BGNY [84]; State-exploration (model-checking techniques), e.g. NRL [259], FDR [249], AVISPA [336], Athena [326]; and Theorem-proving, e.g. Paulson's inductive approach [286] and Coral [328]. The model-checking tools are fully automatic verification tools that are capable to determine whether a protocol satisfies or fails to satisfy a security requirement. In the latter case, the model-checking tool outputs a counterexample of violating that requirement. A *counterexample* is an interleaving of one or more protocol runs violating a given security requirement. We call it an *attack* for short and simplicity.

The research area of communication and security protocols is developing dynamically very fast. Many protocols have been developed and others required to be developed to address the existing weaknesses and the new challenges. This has led the research community to develop good practices for protocol design and to search many useful techniques for protocol verification. The seminal work of Abadi and Needham [20] provides a set of good practices for protocol design. These principles capture common features among protocols that Abadi and Needham found hard to analyze. If these features are avoided, they concluded, protocols tend to become more readable and, more importantly, correct. Other than protocol verification, there are application aspects of formal methods such as: protocol synthesis [292, 122] and diagnosis and repair [214, 297].

2.2 Background

2.2.1 Cryptography

In daily life, we always keep our money, personal IDs, bank cards, etc, safe. This is also true for other valuable items that needed to be protected against malicious actions, we would keep them inside a safe box with a sophisticated lock. Security in communication systems is similar to this example.

Figure 2.1: Some Bikes need to be very secure [1]

In network security, the box, in the example, is the *cryptographic system*. Cryptographic system goal is to provide confidentiality of the communicated messages. This is achieved by altering the *plain* messages following a pattern or key so they become incomprehensible. The process of converting *plaintex* messages into *ciphertext* is called *encryption* or enciphering. The ciphertext message can be recomposed following the same pattern or the inverse of the pattern. The later process, is called *decryption* or deciphering process.

Traditional cryptographic algorithms rely on a fact called "*security through obscurity*" which means that the ciphered process is secret. This old concept is still used.

The appearance of *modern cryptography* was along with appearance of computers. The *rotor machine* ***Enigma*** (Figure 2.2) was the most popular rotor machine that was used for ciphering messages by the German army during the second World War (WW2). As an offensive action by the Allied army, a project called *ULTRA* was developed in order to decrypt the Enigma messages. During the ULTRA project, the Colossus computer, one of the first computers in the history, was designed and used.

From then on, the development of modern cryptography is in parallel with progress of communication systems. Furthermore, modern cryptography systems support also financial operations and e-commerce, distributed systems (e.g., grid, cloud, sensor networks), and so on. Also, not like the traditional cryptography that only provides confidentiality , the current cryptography guarantees several security

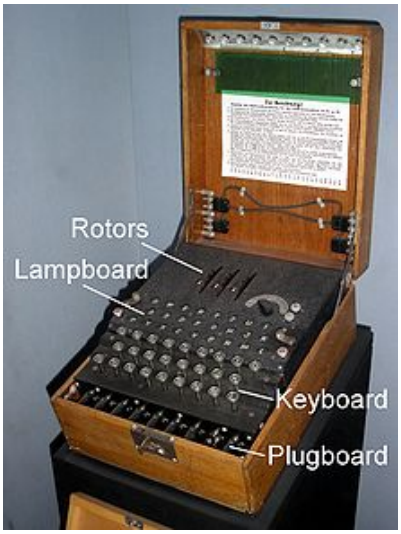


Figure 2.2: Enigma Machine [2]

requirements, i.e.; integrity, non-repudiation, authentication, authorization, and so on. Briefly, we can classify the current cryptographic systems into two groups: (i) cryptography with key (i.e., Symmetric Key and Asymmetric key), and (ii) cryptography without key (i.e., Hash functions)

2.3 Security Properties and Mechanisms

2.3.1 Security Properties

Security properties (or goals) are the characteristics or services of a protocol in order to resist such aforementioned attacks. Several desirable properties have been identified for security protocols, often described with confusing or inconsistent terminology. In verification terminology, the properties of security protocols are expressed usually as *safety* properties (i.e., nothing "bad" ever happens). However, *liveness* properties (i.e., something "good" will eventually happen).

The well known and widely used **CIA**[1] benchmarks for evaluation of information security, focusing on the three core security goals, viz, confidentiality, integrity and availability. Depending on the application, a protocol is expected to provide a subset of these properties. Following, we present an intuitive description of number of security properties and mechanisms.

[1] CIA here does not mean the *Central Intelligence Agency* or the *Culinary Institute of America*.

Confidentiality

Confidentiality refers to protection of resources from unauthorized access or disclosure and limiting access to authorized users. Thus, *who* `is authorized or allowed to access` *which* `resources?`. Confidentiality is related to the broader concept of `privacy` which means limiting access to individual's personal information. *Authentication methods* like user ids and passwords, that uniquely identify system's users, and *control methods* that limit each identified user's access to system's resources, form the basic structure of confidentiality goal. An example of confidentiality violations as; eavesdropping a telephone conversation or reading the emails on one's computer. Possible security threats against confidentiality are disclosure (unauthorized access to information) and usurpation (unauthorized control of resources).

Authentication

The authentication property requires that messages, passing between protocol parties, cannot be forged [316]. That is, a particular message pretending to have originated from a particular source really did originate from that source.

Anonymity

Anonymity is the state of being not identifiable within a set of subjects, the anonymity set [130]. A system that is anonymous over some set of events E should have the property that when an event from E occurs then an observer, though he may be able to deduce that an event from E has occurred, will be unable to identify which [310].

Integrity

integrity refers to the protection of resources from unauthorized manipulation. In security protocols terminology, integrity means that, the content of messages passing between the communicated parties, is not altered by the adversary. Integrity can be achieved using digital signatures (assuming that the used hash function is collision free). Integrity violations like; changing the receipt of a bank transaction or tampering with files on one's computer. Deception (e.g., masquerading) is a possible security threat against integrity goal.

Non-repudiation of origin

Non-repudiation of origin is a safety property provides the recipient with a set of evidences which ensures that the claimed sender has sent the message, even if the sender tries to deny it. This property protects the recipient against a dishonest sender.

Non-repudiation of receipt

Non-repudiation of receipt service provides the sender of the message a set of evidences which ensures that the recipient has received the message. This property protects the sender against a dishonest recipient.

Availability

While availability is typically thought of as a performance goal, it can also be thought of as a security goal. As a security goal, availability refers to the availability of resources or protection of resource from unauthorized *disruption*. An attacker that is interested in reducing the availability of a system typically launches a denial-of-service (DoS) attack. Redundancy, restrictions and load balancing are examples of security measures of availability. As an example of availability attack, an attacker formats your computer's hard disk.

For specification and verification purposes, confidentiality and integrity requirements usually expressed as safety properties and availability requirements are expressed as liveness properties. This process is usually achieved using temporal logics.

2.3.2 Security Mechanisms

To assure security goals and protect them from intentional threats (i.e. disclosure, deception, disruption or usurpation), security efforts or mechanisms are needed. These mechanisms can be divided into those oriented to *prevention* and those focused on *detection* and *recovery*. A security *mechanism* is a method or tool enforcing a security *policy* which states what is allowed and what is not allowed. Following, a brief description of security mechanisms strategies:

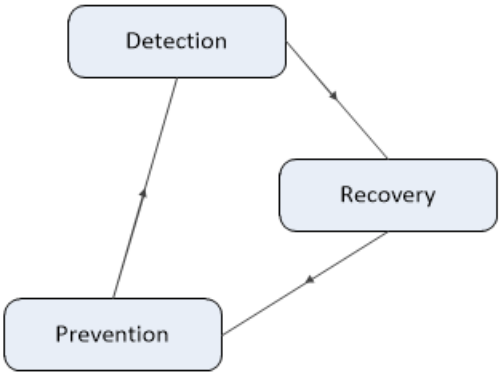


Figure 2.3: Strategies for Security Mechanisms

Prevention of Attacks

Prevention methods aim to protect security goals prior to violation of them. For example, *authentication and encryption*, which restrict access to information and resources. Another example is, *data reduction and separation*, that means removal or separation of information and resources. Prevention mechanisms are, however, not applicable in many settings (e.g. open services).

Detection of Attacks

Where attacks could not be or at least were not prevented, there should be a detection mechanism to detect those attacks. Usually, detection of attacks happens during violation of security goals. *Anti-virus scanners* and *network intrusion detection* are famous examples of detection mechanisms. The major limitation of detection mechanisms is that, they are ineffective against unknown and invisible attacks.

Recovery from Attack

After the violation of security goals and detection of attacks, there should be a recovery mechanisms to bring back the system to its stable state. Thus, recovery from attack happens after violation of security goals. An example, *malware analysis* that deals with observation and analysis of malicious softwares. One drawback of recovery mechanisms is that, a severe damage might have already occurred before the recovery takes place.

2.4 Types of Attacks

The basic classes of threats are: *Disclosure* which means the unauthorized access to information (e.g. eavesdropping), *Deception* means acceptance of false data (e.g. masquerading), *Disruption* that means interruption or prevention of correct operation (e.g. blocking, manipulation), and *Usurpation* which means unauthorized control of resources (e.g. impersonation). An attacker can utilize one or more of the aforementioned threats to launch different kinds of attacks against legitimate participants of a security protocol. An *attack* is the temptation to violate a security goal intentionally (an intentional threat) and often is a combination of different threat classes. Here, we give some examples of attacks:

- *Snooping* or passive eavesdropping of information such as network sniffing or keyboard logging. This can be carried out using disclosure threat class.

- *Manipulation* or active modification of information like control flow redirection and man-in-the-middle attacks. Threat classes deception, disruption and usurpation can be used to fulfill this kind of attack.

- *Spoofing* that is impersonation of one entity by another. Examples of spoofing attacks are; address spoofing and phishing attacks. Deception and usurpation threats can carry out such attacks.

For the verification of security protocols purpose, below we list several types of attacks that can be countered during the analysis of security protocols.

Man-in-the-middle Attack

The intruder imposing himself on the communications between the sender and the receiver. If the protocol is poorly designed the intruder may be able to subvert it in various ways. This attack enables the intruder to masquerade as the receiver to the sender (or vise versa).

Replay Attacks

Replay attacks take place when the intruder redirects eavesdropped or altered messages within one (or possibly more) interleaved protocol session(s). This can happen if the protocol does not have any mechanism for distinguishing between different runs of the protocol or if it has a lack of determining the freshness of messages. Following, a briefly description of replay attacks classification in [333, 53].

- **Reflection attacks:** it is also know as "Mirror attack", has its name from the trick to let a participant answer his own questions. In a reflection attack the intruder resends an altered version of a previously sent message back to its sender. Reflection attacks can be performed within the same protocol session or between contemporaneous protocol sessions, however, it is also possible to reflect messages obtained from previously finished protocol sessions.

- **Deflection attacks:** In a deflection attack the intruder redirects a possibly altered sent message to some participant that is neither the message's recipient nor the sender. Run-internal deflections are performed within the same protocol session. Interleaving deflections use contemporaneous protocol sessions and classic reflections use messages obtained from already finished protocol sessions.

- **Straight replay attacks:** In a straight replay attack the intruder resends a previously sent message to its intended destination. If the eavesdropped message is replaced by an altered version, this attack is also known as integrity violation attack. Also, straight replay attacks are characterized either as run-internal, interleaving or classic replay attacks, depending on whether the attack is performed within the same session or simultaneous or non-interleaved sessions, respectively.

Algebraic attacks

The fact that cryptographic functions often satisfy various algebraic identities. It must not be forgotten that it may also be possible to break the underlying cryptographic algorithms. Almost every known algorithm has one or more algebraic identities (which may be already known or -worse- not yet discovered) which are not necessary for the algebraic purpose but are a consequence of the mathematical structure. A number of examples of these attacks are known, (see e.g. [177]).

Type flaw attack

A type flaw attack arises when the recipient of a message accepts that message as valid, but imposes a different interpretation on the bit sequence than the protocol participant who created it. Type flaw attacks may result in failures of security properties beyond the typical secrecy and authentication properties, like for example anonymity and non-repudiation [201].

Dictionary attack

Dictionary attack is a general threat to all passwords. An attacker who obtains some sensitive password-derived data, such as a hashed-password, performs a series of computations using every possible guess for the password. Since passwords are typically small by cryptographic standards, the password can often be determined by brute-force. This attack is very dangerous because many people tend to choose a poor password. Depending on the system, the password, and the skills of the attacker, such an attack can be completed in days, hours, or perhaps only a few seconds.

Buffer overflow attack

Buffer overflow problems always have been associated with security vulnerabilities. This problem arises due to the bug or programming language or the mistake of programmer, not the design fault of security protocols. For example, a worm called *Slapper* [171, 41, 293] infected 12,000 severs running the SSL module of Apache. The worm does not exploit an attack against the SSL protocol itself, but is a buffer overflow attack.

Denial of Service (DoS) attack

According to [101], "a *denial of service* (DoS) attack is characterized by an explicit attempt by attacker to prevent legitimate users of a service from using that service". Examples of those attempts are; *network flooding* and *preventing or disrupting a system from accessing a service*. There are several types of DoS attacks that aim at a variety of services. The basic type of DoS attacks is the *Consumption of Scarce Resources* (such as bandwidth, CPU and memory consumption).

2.5 Attacker Models

The seminal intruder model, *Dolev-Yao intruder model* [162], was the starting point for security protocol model checking tools. It is based on state space exploration. Dolev-Yao model assumes that the intruder has full control over the communication network. The basic assumptions of the Dolev-Yao model are: (i) the used encryption algorithm is unbreakable (perfect security), (ii) the intruder can stop messages reaching their destination, and (iii) the intruder can create messages of his own. In this approach, the cryptographic primitives are represented as black boxes that follow a set of algebraic rules.

Most of model checking approaches use the general *Dolev-Yao* intruder model, but the intruder model is restricted in finding out a message that is meant to be secret or in generating messages that impersonate some protocol participant. Failures of secrecy or authentication reveal a previously unknown attack on the analyzed protocol. The *Interrogator* [263] and NRL protocol analyzer [259] tools were the first security-purposed analysis tools that were based on the Dolev-Yao model.

A more detailed description of Dolev-Yao attacker model is presented in [57], which is called the "*Lazy Intruder*". This intruder is implemented for the on-the-fly model checker (OFMC) which is one of the four back-ends of the AVISPA tool-set [11]. The lazy intruder is a symbolic approach that represents intruder messages symbolically using terms with variables. The constraints about what terms must be generated and which terms may be used to generate them, are stored and manipulated. This can avoid the explicit enumeration of the possible messages the Dolev-Yao intruder can generate. The resulted symbolic representation is evaluated in a demand-driven way and this approach reduces the search tree without excluding any attacks.

Other contribution for developing intruder models is the work in [53]. The intruder model in [53] adopts the assumptions of the Dolev-Yao intruder, but instead of specifying its behavior with a set of rules governing deducibility of messages, it combines multiple attack tactics based on a careful analysis of how they proceed. Four types of attack tactics have been implemented, namely: (i) Replay and integrity violation attacks, (ii) Type-flaw attacks, (iii) Impersonation attacks, (iv) Parallel session attacks.

However, a class of liveness properties of security protocols (e.g. fairness) which state that some desired situation eventually will occur. The standard Dolev-Yao intruder is not suitable for verification of such requirements because the communication channel is assumed to be under absolute control of the intruder which can destroy the transmitted messages. Cederquist and Dashti [100] proposed a process algebraic intruder model for verifying liveness properties of security protocols. For these types of properties, the authors proved that the proposed model is equivalent to Dolev-Yao model that does not delay indefinitely the delivery of messages.

2.6 Analysis Approaches of Security Protocols

Security of a security protocol depends on the interacting components that conform the environment. In box example in section §2.2.1, it is obvious that if we could open the box, the system is insecure. In a complex network protocol this kind of analysis is more complex.

Because of the fact that safety is a critical issue in development of security protocols, the use of rigorous, mathematical models to develop security protocols becomes recommendable but sometimes mandatory. Using mathematical models helps to identify software problems at earlier stages, prior to coding phase.

There exists two approaches to the verification of security protocols: one is the *formal methods approach* which is based on what can be learned from interacting with several principals engaged in an arbitrary number of protocol runs , and the other is the *computational complexity approach* which is based on the complexity and the probability of breaking the cryptographic primitives of a protocol. These two approaches differ mainly in how they model cryptographic primitives.

Formal Methods Approach

This approach treats cryptographic operations in a purely formal way, for example, the expression $\{M\}_K$ may represent an encrypted message, with plaintext M and key K, whereas, $\{M\}_K$, M and K are formal expressions, rather than sequences of bits [21]. It is assumed that the cryptographic operations are perfect (the *perfect cryptography assumption*). This assumption states, in order to decrypt the message $\{M\}_K$ the appropriate key K should be applied. However, the intruder cannot recover M or K for just the message that is created from the encryption of M under K, represented by $\{M\}_K$.

The intruder is modeled as Dolev-Yao [162] intruder model -so-called *spy*- which is an omnipresent agent that controls the network but cannot make cryptanalysis. However, the adversary can intercept messages, analyze them, and decrypt them if he possesses the corresponding decryption key. He can also inject new messages to the network and send them under any agent name. Additionally, the spy knows all public keys, his own shared key and private key, all shared and private keys of a collection of compromised agents, and a set of lost session keys.

The Computational Complexity Approach

This approach views cryptographic primitives as functions on strings of bits. In [22], Abadi and Rogaway explained an example of the computational approach, they sketched a notation of secure encryption. They defined a symmetric encryption scheme as a triple of algorithms $\prod = (K, E, D)$. Algorithm K is a key generator which after making random choices generates a string k (the key). Algorithm E is an encryption algorithm which flips random coins r to map strings k (the key) and m (the plain-text) into a string $E_k(m, r)$ (the cipher-text). Algorithm D is a

decryption algorithm that maps strings k (the key) and c (the cipher-text) into a string $D_k(c)$. It is expected that $D_k(E_k(m,r)) = m$ for appropriate k, m and r.

The adversary is modeled as Turing machine which has access to an oracle. The oracle has some clues which involve knowledge of some components of m, knowledge of other message encrypted under the same key, etc. The aim of the oracle is to find a key k' that is able to decrypt a given ciphertext $E_k(m,r)$. Roughly, a protocol is considered good if the oracle cannot find k' , or while consuming the computational power at hand the probability of finding k' is slow-growing under a determined threshold. Although providing strong security guarantees, proofs under this approach are in general harder and more difficult to automate.

Recently, Several studies have investigated the connections between the formal view and the computational view. Abadi and Rogaway [22] have bridged the gap between the to views of cyrptography operations by representing two accounts of sysmmetric encryption: one is simple based on formal approach, and the second is more elaberated based on the computational approach. They showed that security properties that can be proved in the formal model are also true the computational model. Later on Baudent *et al.* [59] introduced a reasoning framework for proving soundness of implementations of equational theories, which are used to specify cryptographic primitives. More recently, Kremer and Mazare [223] have extended Baudent *et al.* work to consider an adaptive user rather than a purely passive one. terms of computational indistinguishability. Later on Baudent *et al.* [59] introduced a reasoning framework for proving soundness of implementations of equational theories, which are used to specify cryptographic primitives. More recently, Kremer and Mazare [223] have extended Baudent *et al.* work to consider an adaptive user rather than a purely passive one.

In this thesis, among the two mentioned approaches, we deal only with the formal methods approach for analysis and verification of security protocols. Formal methods have been successfully applied to the analysis of hardware systems, software systems and communication protocols. In the 70's and 80's some initial works appeared to support the analysis of security protocols. In 90's, formal methods have been widely used for analyzing security protocols which allowed to discover some attacks on important security protocols considered to be secure for several years [248].

This chapter discusses the formal methods approach of analysis and verification of security protocols. Before going further of surveying and discussing the most significant formal methods and related tools, we briefly describe the formal methods approach to software verification and give an example of faulty security protocol from academic literature.

2.6.1 Formal Verification of Software Systems

In the context of software systems, formal verification is the act of proving the correctness or the falsification of the intended algorithms underlying the system

with respect to a certain formal specification or property, using formal methods of mathematics. To be able to use the formal software verification, both the system and its specification are first expressed as formulas of some (but not necessarily the same) logic. Then, mathematical reasoning is used to prove that the system and the specification are related somehow, for example by inductive logic. A state-of-the-art verification tool is capable of yielding one of two outputs: (i) OK, indicating that the system is error-free, at least with respect to the coverage analysis of the corresponding tool; and (ii) a counterexample, indicating how a system execution violates the specification.

In the context of security protocol verification, the system is the security protocol under analysis, the specification is the protocol security requirement (security property; secrecy, authenticity, non-repudiation, anonymity, ...) and the counterexample is actually an attack. Authentication and secrecy are the most common examples of protocol security requirements. These properties have no universal interpretation and are formalized according to the context. Roughly, *user authentication* amounts to attempting to verify the identity of a protocol participant and *secrecy* to ensuring that certain message parts sent over the network remain readable only to their intended recipients. Most of the formal methods have been developed and their related tools have studied *secrecy* and *authenticity* intensively, form example [310]. There are also some other work and tools which have studied other properties, for example *non-repudiation* was studied by Judson Santiago and Laurent Vigeron in [215]. For example, Tom Chothia *et al.* in [338], they checked the *anonymity* in a possibilistic general-purpose process algebraic verification tool-set, by using a combination of dedicated tools and the existing μCRL tools. Tom Chothia *et al.* in [338] have studied the Dining Cryptographers problem and the FOO 92 voting protocol, for the proof of concept of their method.

The seminal work of Clark and Jacob [124], so-called Clark and Jacob library[2], and the AVISPA[3] project library documented most of security protocols. Another rich documentation for authentication and key establishment protocols is [83].

2.6.2 An Example of Faulty Security Protocol

To show the difficulties designing sound security protocols, let us consider the Andrew secure RPC protocol [314], which aims to establish a fresh session key between two agents A and B. In the first three messages, A and B perform a handshake using a key they already share, K_{AB}. In the final message, B sends a new session key K'_{AB} to A. Nonce N_A is chosen by A and nonces N_B, N'_B are chosen by B. The so-called *Alice-Bob* notation of the RPC protocol is shown in following list:

Burrow *et al.* [95] have pointed out a major problem with Andrew secure RPC

[2]An online Repository for the Clark and Jacob library is available at http://www.lsv.ens-cachan.fr/Software/spore/index.html

[3]The AVISPA library is available at http://www.avispa-project.org/

Protocol 2.6.1 Andrew secure RPC protocol

1. $A \rightarrow B : \{N_A\}_{K_{AB}}$
2. $B \rightarrow A : \{N_A + 1, N_B\}_{K_{AB}}$
3. $A \rightarrow B : \{N_B + 1\}_{K_{AB}}$
4. $B \rightarrow A : \{K'_{AB}, N'_B\}_{K_{AB}}$

protocol: A has no assurance that K'_{AB} is fresh. An intruder could substitute a previously recorded message 4 (from B to A) and force A to accept an old, possibly compromised, session key. Another problem was pointed out by Clark and Jacob [123]. They proposed a typing attack in which an intruder records a message 2 and substitutes it in place of message 4, as follows:

Attack 2.6.1 Clark-Jacob attack on Andrew protocol (Protocol 2.6.1)

1. $A \rightarrow B : \{N_A\}_{K_{AB}}$
2. $B \rightarrow A : \{N_A + 1, N_B\}_{K_{AB}}$
3. $A \rightarrow B : \{N_B + 1\}_{K_{AB}}$
4. $I_B \rightarrow A : \{N_A + 1, N_B\}_{K_{AB}}$

The result of the attack is that A accepts the value $N_A + 1$ as a session key with B. Clark and Jacob point out that the potential damage of that attack depends on what property the nonce N_A is assumed to have. If N_A is a predictable nonce such as a counter value, then the attacker could force A into accepting a bogus quantity as a session key, whose value could be known to the attacker. If N_A were random, however, the potential damage of the attack is not so immediate since there is no release of the session key.

In the next section, we shall focus on the formal methods for analyzing security protocols.

2.7 Formal Approaches for Verification of Security Protocols

2.7.1 An Overview

Encryption is a useful mechanism that can be used in the construction of secure systems, but it could be applied in the wrong way. It is not a guarantee for security by itself.

Security protocols are a means to ensure some form of secure communication, and they usually try to establish this by using some form of encryption. Security protocols underlay the current communication applications such as; secure Internet communications, cell phone networks, ATM machines, and so on. For such applications, it is crucial that no malicious party can disturb the intended workings of the protocol, or eavesdrop on something he was not supposed to hear. It is not sufficient to have a security protocol of which one is pretty sure that it is secure. Instead, we want some guarantees about its security.

In order to reason about the security guarantees of a protocol, we have to create a mathematical model of both the protocol and the network, which is under the control of an adversary. Such a model allows us to prove e.g., that the adversary is not able to disturb the protocol or learn any secrets. As these models already become complex for simple encryption schemes, it is only feasible to reason about the security of full protocols by abstracting away from some (cryptographic) details. Consequently, the need to reason about security protocols has led in 1981 Dolev and Yao [162] to introduce an idealized abstraction of encryption and computer network with three main properties, the first two abstractions concerning encryption and the next concerning computer network:

(i) Cryptography is assumed to be perfect: a message can only be decrypted by somebody who has the right key (there is no way to crack the scheme).

(ii) Messages are considered to be abstract terms: either the intruder learns the complete message (because he has the key), or he learns nothing.

(iii) The network is assumed to be under full control of the adversary. He can remove sent messages and examine their contents, insert his own messages, or reroute or simply transmit messages.

Together, these three properties are known as the Dolev-Yao model: *cryptography is perfect, messages are abstract terms, and the network is under full control of the intruder.*

Given a security protocol, it is possible to develop mathematical techniques to derive security properties of a protocol under the Dolev-Yao assumptions. The result of Dolev and Yao work has been responsible for a branch of research which can be roughly described as *black-box security protocol analysis*. However, building the three properties sketched above into a precise mathematical model, with clear assumptions, and clearly defined security properties, has proven to be a hazardous task.

The defining example for this area of research illustrates some of the subtleties. It concerns the famous Needham-Schroeder public key protocol from [277], published in 1978. The basic version of this protocol consists of three messages that are exchanged between two partners. It is intended to provide authentication for both agents. It was assumed to be correct for over 20 years; now it is or is not correct,

depending on the situation in which it is used. The reason for this change is not to be found in more powerful analysis methods. Instead, it is the assumptions about the intruder that have changed.

In 1989 Burrows, Abadi and Needham published their ground breaking paper [95] on a logic for authentication (which became known as BAN logic), which also depends on the same black box assumptions as the Dolev-Yao model. Using this logic, they were able to prove that several protocols satisfy a form of authentication. Among these protocols was the Needham-Schroeder public key protocol. In 1995, Gavin Lowe [248] claimed to have found an attack on the protocol. It turns out that Lowe's attack required a stronger intruder than the one Dolev and Yao originally had in mind. Around 1980, networks were considered to be something that was used by honest users: attacks would come from the outside. Thus, the intruder was not one of the regular users of the system. During the nineties, this view of networks changed. Now, many large networks were used by users, which were not necessarily trusted. Lowe's attack requires that the intruder is, or has compromised, a regular user. As a result, the intruder model was modified, and the intruder was assumed from now on to control a number of regular users of the system.

After Lowe's breakthrough, a large number of security protocol formalisms and tools have been developed.

In recent years, the area of security protocol analysis and verification has seen explosive growth with numerous formalisms. Schneider *et al.* [310] broadly categorized these approaches into four main categories:

- logic-based
- model-checking, state exploration
- proof-based
- cryptographic (provable security)

There has been trends to combine them. For example, model-checking and proof-based techniques can be combined together. Similarly, some work attempts to bring together the strengths of formal and cryptographic techniques. Following, a survey of several research works in formal methods for analyzing security protocols based on the classification of methods that was given in Section §1.1.

2.7.2 Modal Logic Methods

Modal logics consist of a language to describe various statements of a protocol such as what participants know or believe, and some inference rules which are used to derive new statements from the current ones. The goal of the analysis is to derive a statement that presents the correctness condition of the protocol. The designer's inability to derive it indicates that the protocol may not be correct.

The belief logic approach was the first attempt to automate the verification of security protocols. The BAN logic of Burrows, Abadi and Needham [95] was

developed in 1989 and is one of the first formal protocol verification approaches. BAN aims to formalize what an agent may infer from the messages it has received. BAN allows short, abstract proofs but it is not able to identify a wide variety of protocol flaws. This is because BAN does not consider an intruder on the network. Its main advantages are: proofs in BAN logic are simple, short and can be obtained manually, and it treats time stamps at an appropriate abstract level. The main disadvantages of BAN logic are: it assumes that agents are all honest, and its limitation on *authentication*. The Needham-Schroeder protocol was proven correct in BAN logic because the man-in-the-middle attack could not be modeled.

The syntax of BAN covers three primitive objects that are principals, keys and nonces. Protocol messages are expressed as formula of the logic.

Proofs are based on deduction rules read as "if formula $X1, \ldots, Xn$ hold then consequently Y holds", written more concisely as:

$$\frac{X_1, \ldots, X_n}{Y}$$

To address its weaknesses, BAN has been extended resulting in new logics, e.g. GNY [187], and Brackin's HOL extension of GNY [84]. These new methods though have proven to be very limited to deal with the general problem of protocol verification and are far beyond from BAN's simplicity. In this domain of formal methods, there some works aimed to automate some of these logics, for example, the work of Mathuria *et al.* [255], which aimed to automate the GNY logic presented in [187].

2.7.3 State Exploration Methods

A protocol is characterized as the set of all its possible traces. Given an input security protocol, the verification method explores as many execution paths of the protocol as possible, checking at each state reached if some conditions hold. The search space generated from analyzing a cryptographic protocol may be infinite, because protocol participants may engage in a number of protocol runs, simultaneously play different roles (initiator, responder, etc.) and because the adversary is capable of building an infinite number of messages. The state exploration approach, however, presents a severe limitation: keeping a state space manageable imposes drastic simplifying assumptions in the formalization of a protocol. There are many reduction techniques developed to cope with the sate explosion problem such, partial-order reduction [128, 127], symmetry reduction [129, 166], symbolic representation of state space using BDD [89, 91, 140, 71], and so on.

Yet, in order to avoid exploring the entire state space, most of state exploration techniques use theoritical results, for example, such as in [332] and [309]. State space exploration techiques are powerful and automatic. They can easily build a counterexample from the generated traces if a property does not hold.

The Dolev and Yao's approach [162] of verifying security protocols was the first attempt at using state exploration for verifying security protocols. Dolev and Yao's approach is however extremely limited, because it considers only the verification

of secrecy and a few number of cryptographic primitives. Dolev and Yao main contribution is a formal model of the intruder, which has been used largely after minor extensions.

In what follows, we will present a survey of the most important state exploration techniques and tools based on model checking and strand spaces.

2.7.3.1 Model checking

Model checking methods consider a relatively large, but finite, number of possible protocol behaviors (the state space) and allow that they satisfy a set of properties. The state space is explored using an algorithm to find out a path which leads to a state where the properties are not held. Model checking methods are generally more suitable for finding attacks on protocols, rather than proving their correctness.

A survey of the most important model checking approaches is given in the following sections.

CSP approach

Gavin Lowe [249] has developed a method for verifying security protocols using FDR, a model checker for process algebra CSP [307]. This method has been used to find a previously unknown attack on NSPK. To verify a security protocol using Lowe's method, each principal taking part in the protocol is modeled as a CSP process representing the protocol steps performed by the principal. In CSP, communication is an event modeled by the notion of *channels* as the form $c.v$ where c is the name of the channel on which the communication takes place and v is the value of the message that passes through the channel. Then, the standard CSP theory of traces is used to conduct the protocol analysis. To check whether a given property holds for a protocol, one test for refinement between the CSP processes representing the protocol and the property in question. In CSP, a process P refine process Q if every trace[4] of P is also a traces of Q.

The task of producing a CSP description of the protocol to verify is very time consuming and requires a high level of skill. To get around this situation, Lowe developed Casper [250]. Casper translates the Alice and Bob description of a security protocol into a CSP model, suitable for FDR. While Lowe results were encouraging, it was clear then that special-purpose tools were required for the verification of large security protocols to be possible. A comprehensive introduction to the method including background on CSP, is contained in the book of Ryan and Schneider [310].

Murϕ

Murϕ (Murphi) is general-purpose model checker with its own language. It has been used to verify several kinds of protocols and algorithms over several years.

[4] A trace is a sequence of events

Similar to other model checkers, the idea is to specify certain security properties and then search all reachable states and check whether the properties are always satisfied. The reachable states are limited by the number of instances of each principal (including the adversary) that are modeled and the number of protocol messages.

Mitchell *et al.* [209] used Murϕ to analyze three establishment protocols. The three protocols are Needham-Schroeder public key protocol, the TMN protocol and the Kerberos protocol. In each case the tool was able to quickly find previously known problems. Later, Mitchell *st al.* [272] used Murϕ to analyze the widely used SSL (SSL 3.0) protocol. Although no attacks were found which revealed keys or violated authentication, some anomalies were uncovered by the analysis. None of these anomalies poses a direct threat to the security of SSL 3.0.

Mitchell *et al.* [209] reported that protocols with three to five messages and four to five principal instances could be analyzed in a few minutes, but doubling these numbers could lead to an analysis requiring several hours and possible memory problems.

The methodology of an analysis with Murϕ is similar to that using FDR. Mitchell [209] identified two main differences between the two tools.

1. FDR has an explicit notion of channels for communication between processes, whereas Murϕ models communication through shared variables.

2. Murϕ has a richer set of methods to reduce the searching time required for a given choice of protocol parameters.

The intruder model used with Murϕ has usually taken a simple *"perfect black-box"* view of cryptography which does not distinguish authentication from encryption or recognize different types of confidentiality.

Brutus

Brutus is a special-purpose model checking tool for security protocols. According to Clarke *et al.* [126] it was developed to allow *"push-button"* checking of protocols so that designers easily obtain the assurance of formal analysis. If a problem is found with the protocol then Brutus will provide an explicit attack.

Clarke *et al.* gave a sample analysis of three protocols, two of which were for key establishment. One of these was the Needham-Schroeder public key protocol, for which they were able to rediscover the flaw previously found by Lowe using FDR. Another was the Wide-mouthed-frog protocol, however, because their model does not include a notion of time, they were unable to find known replay attacks.

As the same in other model checkers, security protocols are modeled in Brutus by describing the operational behavior of agents or principals participating in the protocol. Messages are formed from keys, principal names and nonces, as well as any data to be conveyed. An assumption of *perfect encryption* is made, which requires that a ciphertext $\{m\}_k$ can only be formed with knowledge of key k and message m. The set of messages known to any principal is recorded.

The logic was a variant of the linear-temporal logic with the past-time operator where at the atomic level one can express actions and knowledge. This flexible logic allows many protocol properties to be specified suitable for different requirements. For example, for the Needham-Schroeder protocol, three properties are checked. One is an *authentication* property which demands that if an instance of A has completed a protocol run apparently with B, then an instance of B must at lease starts a protocol run with A. The second property is *secrecy* which requires that if the adversary knows a nonce of principal A then A must be running the protocol with the adversary. The third requirement is a *weak non-repudiation* property which ensures that if an instance of A has finished a protocol run with B then it must know the nonce of A. The analysis found that the protocol satisfies only the last of these properties.

Clarke *et al.* [126] provided an extended comparison between Brutus and other formal approaches (logic of authentication, strand spaces, Isabelle, FDR, Murϕ, ...) to analysis of security protocols. They stressed that the main benefit of their tool is the ease of operation for the partitioner, while the special-purpose language is more expressive than some general-purpose approaches. They also described extensive measures to reduce the complexity of the search process.

Lazy Intruder Approach

In 1999, Basin [58] presented an an approach for analyzing security protocols that combines complementary aspects of model checking and interactive verification. The key idea to deal with state explosion problem and to model possibly infinite state space of the protocol. Basin used *lazy data types*. A lazy data type is one where data-type constructors build data types without evaluating their arguments, thus allowing for representation and computation of infinite data in a demand driven manner.

The semantic formalism that Basin has used for modeling protocols and attackers is a trace based interleaving semantic, motivated by, and closely following, Paulson's [286] work. Rather than formalizing protocols as inductively defined set as Paulson, Basin formalized them as infinite trees. The nodes of the tree are traces and children correspond to trace extensions by one step of (some run of) the protocol or an action by an attacker (Dolev-Yao spy). Hence, a protocol, along with an attacker model, defines an infinite tree and a security property is a property of nodes in the tree. Violations of security properties are found by a kind of infinite-state model checking, which is performed by searching the tree in a lazy fashion. If there is an attack, it will be present in a trace located at some node in the tree. Finding this node, however, may not be easy: not only may the tree be infinitely deep, but it may also result in an exponentially large branching factor.

With a large branching factor, standard search algorithms would hardly succeed in finding attacks that lie at shallow depths. To get around this problem, Basin used two simple heuristics, one for pruning the search tree and the other for reordering the way in which the tree is searched. The first heuristic consists of pruning traces

that contain spurious events, such as the reception of a message that does not obey the rules of the protocol. The second heuristic consists of assigning the highest priority to events involving the start of another execution of the protocol or the execution of an action by the spy.

OFMC approach

Basin *et al.* in [56] presented the On-the-Fly Model Checker (OFMC), a tool that combines two ideas for analyzing security protocols based on lazy, demand-driven search. The first is the use of lazy data types as a simple way of building efficient on-the-fly model checkers for protocols with very large, or even infinite, state spaces. The second is the integration of symbolic techniques and optimizations for modeling a lazy Dolev-Yao intruder whose actions are generated in a demand-driven way. OFMC's methods significantly reduce the search space without excluding attacks. In order to validate their approach (OFMC), the authors carried a large number of experiments. They found that OFMC is able to rediscover all known attacks, and discovers a new attack (on Yahalom protocol), in test suit of 38 protocols from Clark-Jacob library [124] in a few seconds of CPU time for the entire suite. OFMC was also able to discover attacks on large-scale protocols such as H.530 protocol [195] as described in [56]. OFMC is implemented as one of the AVISPA's back-ends. One drawback of OFMC is that it cannot verify group security protocols.

The formal model that OFMC uses for protocol analysis is based on two specification languages: a high-level protocol specification language (HLPSL) and a low-level one Intermediate Format (IF). These languages were developed in the context of the AVISPA project [3].

Constraint-logic based approach

CL-Atse is an *OCaml-based* implementation of the deduction rules developed in the CASRUL tool-suite and the AVISPA European project. These rules allows anyone to interpret and automatically "execute" a protocol in every possible ways against a generic intruder with the Dolev-Yao deduction capabilities. A protocol, written in the Avispa's intermediate format, is executed over a finite (user decided) number of iterations, or entirely if no loop is involved. There are no other restrictions: either an attack is found, or the protocol is claimed to be secured over the given number of sessions. CL-Atse was developed based on the work of Jacquemard, Rusinowitch and Vigneron [210] and the work of Chevalier and Vigneron [354, 113]. The CL-Atse combines rewrite-based first order theorem proving with constraint logic in order to handle properties such as associability/commutativity of operators for representing sets of messages. CL-Atse performs two kind of optimizations : i) the protocol specification itself is simplified to accelerate the search for attacks (steps merged or removed, useless messages ignored, etc). ii) various optimizations in the deduction rules try to reduce, and often eliminate, redundancies and uselessness in the protocol execution.

Rewriting and tree automata approximation approach

In 2000, Genet and Klay in [182] presented an approach for verifying security protocols based on term rewriting systems and tree automata approximation techniques. The used approximation method is an extension of the Genet's work detailed in [181]. Genet and Klay supposed the intruder is the network (this assumption is a bit stronger than the one that assumes that only the intruder has a full control over the network). A direct consequence of this choice is that the knowledge of the intruder and every message that the intruder can build is supposed to always remain on the network. Furthermore, they supposed that any agents that is not A or B may be dishonest and deliberately give to the intruder their private keys as well as the contents of any message they send or receive.

The main idea of Genet and Klay [182] is to build an automaton whose language represents an over-approximation of the network's configuration with an unbounded number of sessions. If the automaton A_0 represents the initial configuration of the network and the different actions the intruder is able to do (composition of messages and encryption). The intruder abilities to analyze messages and each step of the protocol are defined by TRS. For a given initial configuration A_0, a TRS and approximation function γ, an automaton A_k is computed.

The language recognized by A_k represents an over-approximation of the set of messages that the intruder may know. A tree automaton A_{secret} represents the set of secret terms. To check whether the intruder may know a secret, the intersection between $L(A_{secret})$ and $L(A_k)$ on tree automata is computed. If the intersection is empty then the protocol is safe, otherwise one cannot conclude.

An interesting aspect of this method is that it takes advantage of theorem proving and a form of abstract interpretation called approximation. The basic deduction mechanism, coming from the domain of theorem proving, provides some simple and efficient tools -tree automata- to manipulate infinite objects. On the other hand, approximation simplifies the proof in such a way that it can be automatically computed afterwards.

In 2004, Ohsaki and Takai in [280] extended the work of Genet and Klay [182] to AC-tree automata (tree automata using associative and commutative symbols). The latter work was implemented in ACTAS tool [280].

Another extension of the work of Genet and Klay [182] was developed by Boichut *et al.* [77], which make the approach efficiently automatic. Boichut's *et al.* was implemented in the TA4SP tool with a high level specification language as input format. TA4SP is integrated in the AVISPA tool suite. TA4SP checks only secrecy property of security protocols, it approximates the intruder knowledge by regular tree languages and rewriting. For secrecy property, TA4SP can show if the protocol is flawed (by under-approximation) or whether it is safe for any number of sessions (by over-approximation).

Symbolic approach

Symbolic representation of state space have been developed as a solution of *state explosion* problem instead of using the most obvious way to store the state space in memory which is to use a representation that *explicitly* stores the states and the transitions between them. Typically data structures such as linked lists and adjacency matrices are used. Symbolic model checking techniques use *Boolean formulas* to represent sets of states and transition relations. Traditionally, symbolic model checking has used *Binary Decision Diagrams* (BDD) [90], a canonical form of representing Boolean formulas. The idea of the BDD (*Binary Decision Diagram*) technique is to replace the explicit state space representation by a compact *symbolic* representation.

Many techniques and model checking tools have been developed which are using BDD to represent the state space. The symbolic model checkers SMV [257] and its successor NuSMV1 [119] both implemented BDD-based symbolic model checking.

Since the seminal work of McMillan [257], several mechanisms for a partitioned representation of finite state machines and different exploration styles [94, 135, 302] have allowed to increase the applicability of BDD-based model checking. A new form of symbolic model checking, commonly known as *Bounded Model Checking* [71], has been introduced. Bounded Model Checking is based on the encoding of a model checking problem into a *propositional satisfiability* (SAT) problem, and on the application of efficient SAT solvers. This approach, in the following called SAT-based model checking, relies on the enormous progress in the field of propositional satisfiability [273]. The approach is currently enjoying a substantial success in several industrial fields (see, e.g., [135]), and opens up new research directions.

The work of Biere *et al.* [71] presented a symbolic model checking technique based on SAT procedures. The basic idea is to consider counterexamples of a particular length k and generate a propositional formula that is satisfiable iff such a counterexample exists. This technique is implemented in **BMC** tool based on bounded model checking. [71] gave examples in which SAT-based model checking significantly outperforms the BDD based model checking. This work was the first step in applying SAT procedures to symbolic model checking.

Followed, many model checking tools implemented SAT procedures. One is NuSMV2 [119, 120] integrated both the original implementation of NuSMV1 BDD-based and SAT-based model checking.

A lot of work has been done on SAT-based model checking by Armando *et al.*, particularly during AVISS and AVISPA projects (e.g.; [44, 45, 47, 49, 46]). SATMC is implemented as a back-end of AVISPA tool, It considers the typed protocol model and performs both protocol falsification and bounded session verification by reducing the input problem to a sequence of invocations to state-of-the-art SAT solvers. More specifically, SATMC first builds a propositional formula encoding a bounded unrolling of the transition relation specified by the intermediate format (IF)[5], the

[5]An IF specification describes a protocol in terms of rewrite rules describing an infinite-state transition system with an initial state, transition rules, and a state-based safety property, namely

initial state, and the set of states representing a violation of the security properties. The propositional formula is then fed to a SAT solver and any model found is translated back into an attack.

Strand Spaces

The strand space model is a framework for producing mathematical proofs of security protocol correctness. The technique (like many others) uses the Dolev-Yao model of security protocols, and abstracts the real-world cryptographic algorithms into abstract operations. Hence, Strand Spaces focuses on the structure of a protocol, rather than the protocol's use of any particular encryption scheme. A strand represents the behavior of a principal or the adversary in terms of message sending or receiving.

Originally designed as a pencil-and-paper proof method, it has also been automated. It has been extended in various ways, and its relationship with other proof methods has been explored. It also seems a handy framework with which to prove general results about protocols [4].

Fabrega *et al.* work [167] developed the basic notation of the strand space method. The motivation of this method is that the model checking tools aim to find an attack trace by identifying if there exists a state in the execution of a protocol that violates a given security requirement and they do not proof that a protocol is really correct. A *strand* is a sequence of events that an agent, legitimate or not, may be engaged in. A *strand space* is a set of strands consisting of strands for the various legitimate protocol agents, together with intruder strands. The elements of a strand are called nodes. In the strand space model, the sending of a message M is represented by $+M$ and the receiving is representing by $-M$. A strand is therefore a sequence of sign prefixed messages. For example, the *initiator strand*, for the Needham-Schroeder Public-Key protocol (NSPK), is:

$$\langle +\{N_A.A\}_{PK_A},\ -\{N_A.N_B.B\}_{PK_B},\ +\{N_B\}_{PK_B} \rangle$$

The main idea of proving protocol properties using Strand Spaces is to use inductive principle on *well-founded* sets. Proofs in the strand space model are carried out with respect to structures called bundles. A *bundle* is well-founded set of strands satisfying the following two conditions: (1) every message received by a strand is actually sent previously by some strand present in the bundle and (2) if a node occurs on a strand then all nodes that would have preceded itself on the same strand also occur on the strand. Bundles thus represent individual runs of a protocol.

Many kinds of authentication properties can be proved using strand space framework. For example, the proofs obtained by Fabrega *et al.* [167] for the Lowe-Needham-Schroeder protocol include assurances that the protocol achieves injective agreement on N_a and N_b for both A and B. The injective agreement for the initiator demands that if principal A completes a run of the protocol as

a goal (attack) predicate that defines whether a given state is an attack state or not. An attack trace is then a path that leads from the initial state to an attack state.

initiator with B, then there exists a unique run of the protocol with principal B as responder and both agree on the specific values of the nonces, N_a and N_b. The injective agreement for the responder demands that if principal B completes a run of the protocol as responder with A, then there exist a unique run of the protocol with principal A as initiator, and both agree on the specific values of the nonces N_a and N_b. These agreement properties are proved by establishing that whenever an initiator or responder strand belongs to a bundle, then matching strand exists uniquely in the bundle.

Guttman and Fabrega in [158] introduced two kinds of authentication tests: (i) An outgoing test is one in which the new value v is transmitted in encrypted form, and only a regular participant can extract it from that form. (ii) An incoming test is one in which v is received back in encrypted form, and only a regular participant can put it in that form. These two tests are combined in an unsolicited test and a related method for checking that certain values remain secret. They gave straightforward proofs of security protocols and showed how to use authentication tests as a heuristic for finding attacks against incorrect protocols. They expressed these ideas using strand space formalism. Based on the authentication tests method, the authors suggested a protocol design process. For illustrating this process, they gave an example showing how to modify the Otway-Rees protocol. Guttman [194] used this method to create *ATSPECT*, an Authentication Test-based Security Protocol for Electronic Commerce Transactions, which aimed to offer functionality and security guarantees similar to the purchase request, payment authorization, and payment capture phases of SET.

Strand space framework is the basis of many tools. *Athena*; an automated verification technique for security protocol analysis which was designed by Song *et al.* [326] is based on the strand space model. Athena implemented Song *et al.*'s extension to the strand space model and utilized techniques from both model checking and theorem proving approaches. Athena is fully automatic and is able to prove the correctness of many security protocols with arbitrary number of concurrent runs. Athena includes optimizations to improve efficiency, including the use of a pruning theorems - to prove early that some states do not contribute to the final result- and automatic evaluator of well-formed formulas.

Athena uses a dedicated logic suitable for strand space model which can express various security properties, including authentication, secrecy, and electronic commerce properties. To verify the protocol, Athena first transforms the security property to be verified into an initial sequent which contains an initial state. It then applies a small set of inference rules with certain decision procedures to the states, building a proof tree, until it either completes the proof or refutes a sequent. In the latter case, Athena reports the protocol to be incorrect, and the state of the refuted sequent represents a counterexample, or a successful attack on the protocol.

Athena has been tested on the Clark and Jacob library. One drawback of Athena is that the verification procedure is not guaranteed to terminate. Termination can

be forced by bounding the number of concurrent protocol runs and the length of the messages.

Scyther tool approach

The operational semantics of security protocols defined in [142] form the underlying model of Scyther tool [143]. For describing protocols, Scyther use the *complete characterization* of a security protocol using pattern refinement. This approach is an alternative and modified approach of the works in [161, 159, 160]. The idea of the complete characterization approach is to give a concise representation of all possible traces (or behaviors) of security protocol and not only verify a specific property [145, 146]. The verification algorithm of the Scyther tool is based on the *Athena* [326] tool algorithm.

ProVerif Tool

ProVerif [72] represents a protocol by Horn clauses and can analyze an unbounded number of runs by using over-approximation. Beside Horn clauses, ProVerif accepts also a subset of π-calculus as an input language. The distinguished features of ProVerif tool are: it is able to find attacks that require any number of concurrent protocol runs, it is fully automatic, and it does not significantly suffer from state space explosion. Protocol specification in ProVerif are Horn theories (i.e. Horn theory is a set of Horn clauses). It is important when specifying a protocol in ProVerif to specify the possible ways an attacker can gather more knowledge. The attacker capabilities are modeled based on the Dolev-Yao intruder model. ProVerif uses an abstraction of fresh nonce generation. This makes it able to perform unbounded verification for a class of protocols. Given the protocol specification, ProVerif generates the following possible outputs viz.; (1) the property is false and an attack trace is produced, (2) the property can be proven correct, (3) the property cannot be proved (e.g. when a false attacks is found) , and (4) the tool might not terminated.

Probabilistic Methods

Probabilistic model checking is a formal verification method for analyzing stochastic systems. Security and communication protocols exhibit stochastic behavior. Usually, the specification logics of quantitative properties of stochastic systems allow to compare the measure of executions satisfying these specification with thresholds. Mainly, there are two approaches for solving the model checking problem for stochastic systems with respect to those logics, namely, *numerical approach* and *statistical approach*. Numerical approaches iteratively compute the exact (or approximated) measure of paths that satisfy relevant subformulas of property specification. The statistical approach is based on *simulation* of the system for many runs, and use of *hypothesis testing* to infer whether the samples provide a *statistical* evidence for the satisfaction or violation of the specification. Examples of probabilistic model

checking tools that use the numerical methods are PRISM [298] and MRMC [296]. Examples of statistical model checking tools are VESTA [317, 318] and YMER [355].

However, all those model checkers are general-purpose model checkers for probabilistic systems. Among them, the PRISM model checker has been used for analyzing several security and communication protocols and security problems. For example, A Certified E-Mail Protocol In Mobile Environments [295], The Needham-Schroeder (NS) and TMN protocol [33] (more details can be found in the PRISM model checker website in the case studies section[6]). PRISM is a symbolic model checker and supports several types of probabilistic models including: discrete-time Markov chains (DTMCs), continuous-time Markov chains (CTMCs), Markov decision processes (MDPs), probabilistic automata (PAs), probabilistic timed automata (PTAs) and extensions of these models with costs and rewards. PRISM describes model using a simple, state-based language, based on reactive modules formalism. PRISM also supports various probabilistic temporal logics as a property specification language, including PCTL, CSL, LTL and PCTL*. PCTL is used for specifying properties of DTMCs, MDPs or PTAs; CSL is an extension of PCTL for CTMCs; LTL and PCTL* can be used to specify properties of DTMCs and MDPs (or untimed properties of CTMCs).

Timed Automata Approach

The timed automata was firstly introduced by Alur and Dill [37] for modeling real-time systems. Timing aspects such as *timeouts* and *time stamps* are not considered in the state of the art techniques for analyzing security protocols. Corin et al. introduced, in [137], a method for analyzing security protocols that consider time issues (both timeouts and timestamps) using timed automata and UPPAAL model checker [40].

Spi Calculus

Abadi and Gordon in [18] introduced the spi-calculus. It is an extension of the π-calculus [266] with cryptographic primitives. It was designed for describing and analyzing security protocols, such as those for authentication and for electronic commerce. Cryptographic primitives and communication through channels are the main ingredients of the spi-calculus.

The behavior and properties of security protocols can be modeled at the high level of abstraction in the π-calculus. In particular the secrecy property can be captured in terms of the π-calculus's scope rules. Additionally, setting up a new channel can be captured by scope extrusion. However, the π-calculus does not express the cryptographic primitives that are commonly used for implementing channels in distributed systems: it does not include any constructions for encryption and decryption, and these do not seem easy to represent. To be able to reason about

[6]PRISM Case studies: http://www.prismmodelchecker.org/casestudies/index.php

cryptographic primitives Abadi and Gordon extended the syntax and correspondingly the semantics of the π-calculus to express cryptographic terms.

The framework can handle various cryptographic primitives: symmetric encryption, asymmetric encryption, cryptographic hash functions and signatures. One pleasing feature of this approach is the way the intruder is modeled simply as an arbitrary environment able to perform any test definable within the language to try to distinguish instantiations of the protocol. This avoids the need to define explicitly the intruder's capabilities and so avoids dangers of missing capabilities. On the other hand, limitations in the intruder capabilities are implicitly coded into the expressiveness of the framework and in particular the richness of the space of tests that can be constructed. It could be argued that for some applications it is better to have an explicit representation of the intruder's capabilities. This allows it to be evaluated more directly and tailored to specific scenarios, for example to situations in which only passive and no active attacks are possible (maybe on certain channels).

Abadi and Gordon have analyzed many authentication protocols existing in literature using both the π-calculus and the spi-calculus. They have shown how to represent the protocols, how to express their security properties, and how to prove some of these properties.

Later in 1999, Abadi in [12] developed principals and rules to achieve secrecy properties. The approach was based on classification techniques by extending these techniques to handle concurrent processes that use shared-key cryptography. The rules have the form of typing rules for the spi-calculus. They guarantee that, if the protocol typechecks, then it does not leak its secret inputs.

The theorems that are obtained depend on the model of the spi-calculus that would be chosen, which is fairly accurate and expressive, but does not take into account issues of key length, for example. The principles and rules developed in [12] are incomplete and not sufficient, because they ignored all security issues except secrecy, and because they did not show how to implement the spi-calculus while preserving secrecy properties.

Several environment-sensitive bisimulation techniques have been proposed. Bisimilarity is an observational equivalence, based on the idea of an environment observing a pair of processes to see whether it may distinguish one from the other. The environment typically observes the labeled transition system derived from the operational semantics of processes. In most process calculi, the two points of view of an observing environment and of an observed process are symmetric: any transition that a process can do according to its semantics is also observable by the environment. This symmetry is no longer valid in the case of the spi-calculus. As a means of capturing the environment knowledge, the notion of *environment-sensitive* bisimulation has been developed for spi-calculus in various styles. Examples of such techiques are [19, 267, 86, 79, 165, 80, 85, 337].

Additionally, spi-calculus continued be extended using different techniques. For

example, in [197] presented and an extension of spi-calculusto iinclude pattern-matching as a primitive. Furthermore, several extensions based on symbolic analysis have been explored such as [205, 39, 78].

Several other process calculi which are extensions of spi-calculus have been developed. These include; Applied π-calculus [17], Probabilistic Applied π-calculus (PAPi) [191], Ambient calculus [98, 237, 92, 93, 320], Crypto-loc calculus [74, 74], and Timed spi-calculus [196].

In [23], Abadi and Blanchet developed two language-based techniques for analyzing security protocols. One is based on type process calculus for expressing and proving secrecy properties of security protocols with generic treatment of many cryptographic operations. And the other one is based on untyped logic programs. The relation between the previous two different techniques (typed system and untyped system) was studied and it was shown how to relate the proof methods in those two techniques.

Blanchet in [73] extended these methods to handle the authenticity properties. The specification of the protocol is described by a process calculus which is an extension of the π-calculus, then the protocol is specification automatically translated into a set of Horn clauses. Finally, these set of Horn clauses is passed to a solving algorithm based on resolution rules. One limitation of the solver is that it does not always terminate. This method have been used to analyze many protocols including; NSPK, NSPK-corrected, Woo-Lam public key protocol, Woo-lam shared key, Woo-Lam shared key corrected, simpler Yahalom, simpler Otway-Rees and main mode of SKeme.

Many researches have introduced methods that used π-calculus or one of its extensions like spi-calculus and applied pi-calculus to describe security protocols. For example, Gordon and Jeffery defined a type system for verifying authenticity in security protocols, first for shared-key cryptography [189], then for public key cryptography [190]. In both cases, the protocol is specified in the spi-calculus. Another example, Abadi and Blanchet introduced an automatic tool for formalizing and verifying a cryptographic protocol for certified email [13]. In [14], Abadi, Blanchet and Fournet presented the analysis and verification of Just Fast Keying (JFK) protocol in the applied pi-calculus with the assistance of an automatic protocol verifier.

2.7.4 Theorem Proving

Theorem proving based methods consider *all* possible protocol behaviors, and allow checking that they satisfy a set of correctness conditions. Theorem proving attempt to produce a formal proof, given a description of the protocol, a set of logical axioms, and a set of inference rules. Theorem proving approaches fall into two categories. One is based on First-Order Logic (FOL) and the other based on Higher-Order Logic (HOL).

2.7.4.1 First-Order Logic methods

Many approaches have been developed for verifying security protocol in first-order logic (FOL). Some of these approaches used a general automated FOL theorem provers (e.g. SPASS[7]) while others have developed their own automated verification tools. Here we investigate, briefly, some of these approaches.

Weidenbach [348] aimed to combine the benefits of the finite state analysis and the inductive method. With this aim, Weidenbach used a fragment of FOL that are expressive enough to have infinite, inductive models, but that are still subject to automated theorem proving. The protocol and its requirements are both translated into first-order monadic Horn fragments. Then, these formula are input to the automated first-order theorem prover *SPASS* [349]. Now, if SPASS terminates that means the protocol is safe. Otherwise the protocol is faulty. In the latter case an attack trace can be built form SPASS's output by hand (this process has not been automated).

Using this process Weidenbach analyzed the Nuemann-Stubblebine protocol and found new flaws in it and he proposed a possible solution. This approach has some drawbacks: it does not automatically generate counter-examples, the intruder model used is weaker than Dolev-Yao's model, and it is not as expressive as Paulson's inductive approach [286].

Cohen [132, 133] proposed a method for verification of security protocols based on first-order invariants. Suitable invariant can be constructed automatically form most protocols, allowing safety properties to be proved with first-order reasoning. Cohen has implemented his method in an automatic verifier, TAPS, that proves safety properties roughly similar to those proven in Isabelle verifications. Protocols are modeled as a transition system, where the state of the system is given by the set of transitions that have been executed and the set of messages that have been *published* (i.e., sent in the clear). A typical transition generates some fresh values (to be used as nonces or keys), checks that some precondition holds, records that the transition has taken place, and publishes a new message. Transitions are also used to model the actions of the Spy. The states of the system can be further restricted by user-supplied axioms.

TAPS takes, as input, a protocol description to generate a suitable invariant and proves it using a resolution prover. Finally, TAPS proves the desired protocol goals from the invariant, again using resolution. TAPS used to verify about 80 protocols, including almost all of the Clark & Jacob Survey. TAPS verification seems to require considerably less user guidance and time than equivalent Isabelle verification. One downside of Cohen's method is that it does not generate counterexamples because it searches for proofs, not attacks.

Steel and Bundy' method aimed to find counterexamples to universally quantified conjectures in first-order logic. Steel and Bundy used the proof by consistency strategy to guide a search for a counterexample and a first-order theorem prover to perform a concurrent check for inconsistency.

[7] Synergetic Prover Augmenting Superposition with Sorts.

Steel *et al.* in [328] described a method for finding counterexamples to inductive conjectures and how this method will benefit the verification of security protocols. Steel and Budy in [327] presented CORAL, a tool for finding counterexamples to incorrect inductive conjectures and they described how they used this tool to model protocols for both group key agreement and group key management, without any restriction in the scenario (number of agents, number of sessions, which agent will play the initiator and which agent will play the responder, and so on). CORAL is a full implementation of the Comon-Nieuwenhuis method for proof by consistency [134]. Proof by consistency is a technique for automating inductive proof. It has also been called *inductionless induction*, and *implicit induction*, as the actual induction rule used is described implicitly inside a proof of the conjectures consistency with the set of hypotheses. Recent versions of the technique have been shown to be *refutation complete*, i.e. are guaranteed to detect non-theorems in finite time. CORAL is based on an adapted version of the theorem prover SPASS [349].

CORAL model is closely related to Paulson's inductive model, as it uses a first-order version of Paulson's inductive model for protocol verification. A protocol is modeled as the set of all possible traces, i.e. all possible sequences of messages sent by any number of honest users under the specification of the protocol and, additionally, faked messages sent by the intruder. A trace of messages is modeled as a list. The intruder model used in CORAl method is the usual intruder model following the Dolev-Yao model.

CORAL model has been tested on many protocols and was able to rediscover several known attacks on two and three party security protocols including Needham-Schroeder, Neumann-Stubblebine and BAN Otway-Rees. More than that, CORAL was able to discover 6 previously unknown attacks on three group protocols: 3 on the Asokan-Ginzboorg protocol, 2 on Taghdiri and Jackson's improved version of the Tanaka-Sato protocol, and 1 on the Iolus protocol [327].

There are two weaker aspects of CORAL's performance. One, is the difficulty to pose conjectures and the second, Coral is slow to find counterexamples because the search spaces created by CORAL's open model are always going to be large, in order to solve this problem some heuristics need to be added. Thus, CORAL does not compete against model checking techniques tools for attack discovery in two or three party protocols since these tools are faster. In this matter, the advantage of CORAL over model checking tools is that CORAL searches for a counterexamples in the general model of a protocol, not just in a model involving particular principals being involved in particular sessions as specified by the user.

Delaune *et al.* in [150] introduced the rigid clauses to model security protocols when the number of sessions is fixed. Then they designed a resolution-based decision procedure for solving such problem and they showed termination results.

Affeldt & Comon-Lundh in [28] used the rigid variables to give formulation of security problems for a bounded number of sessions, that avoiding false attacks. They showed that there is a simple translation of rigid variables in first-order logic, which preserves the satisfiability of the formula.

2.7.4.2 High-Order Logic methods

The inductive approach to the verification of security protocols was invented by Paulson [286] using the proof tool called *Isabelle* with HOL formalism. This approach combines: (i) the concrete notation of events which is borrowed from state exploration method, and (ii) the idea of deriving guarantees from each message which is borrowed from belief logics . Induction is a familiar proof technique in mathematics. If $F(n)$ is a formula to be proved true for each integer n. This requires a base case, $F(0)$, to be established along with a general result that $F(i) \Rightarrow F(i+1)$. Then, it can be concluded that $F(n)$ is true for any positive integer n. The idea of an inductive proof is to identify all desired properties P of a protocol and show that P is preserved under all possible extensions of a given set of of observed events. Thus, this approach is capable of stating whether protocols are insecure. However, it does not explicitly show an attack. This drawback makes the inductive approach impracticable for improving protocols in cases where they are stated as being insecure. Another drawback is that the process tends to be significantly longer and requires more effort.

As with most other formal specifications, messages are made up of principal names, nonces and keys. Messages can also be concatenated to form compound messages. Cryptography is limited to two functions: hashing and encryption. Encryption assumed to be *perfect*.

Three operations are defined on any set of messages H.

- *parts* H represents the set of components than can be recovered from H, which may require knowledge of certain keys.
- *analz* H represents the set of components that can be recovered from H without knowing any secrets (apart from what may be already in H).
- *synth* H is the set of messages that can be formed from H by adding principal names, concatenating message components, or encrypting with keys in H.

A protocol is inductively defined as a set of (all possible) traces. A *trace* is a list of communication events. An *event* refers to either sending a message or receiving a message. The adversary observes all traffic in the network - the set H- and sends fraudulent messages drawn from the set *synth(analz H))*. The adversary is allowed to start or continue any run of the protocol between any principals. A protocol specification describes how the set of observed messages may be extended by adding messages that would be output by principals who received messages which the adversary can read or synthesize from H. The adversary may also be allowed to obtain old session keys.

Paulson [286] described proofs for three protocols: Otway-Rees protocol, Needham-Schroeder public key protocol and a recursive version of the Otway-Rees protocol involving multiple entities. Later Paulson provided an extended analysis of Yahalom protocol [285] and proofs for simplified version of the TLS protocol [287].

Bella in [62] scaled up Paulson's inductive approach for the analysis of classical cryptographic protocols towards real-world protocols. Bella's inductive approach extended Paulson's approach with new elements (e.g. timestamps and smart cards), new network events (e.g. message reception) and more expressive function (e.g. agents' knowledge). Many protocols are analyzed using Bella's method for example Kerberos [64, 65] and Shoup-Rubin protocol based on smart cards design was also analyzed [63].

The inductive approaches developed by Paulson and Bella have been used successfully to analyze Internet protocols , e-commerce protocols and smart card protocols. The downside of these theorems is that they are very complex and difficult to master and they require that the user must guide the proof process and select the tactic to be applied. Juan *et al.* in [213] introduced a method, which based on the idea of belief logic, simplified the description of protocols and proof procedures under inductive approach. The main protocol property (e.g. authentication) is formulated based on belief logic. If these goals are proven then they can be combined to show why the protocol achieves security. Otherwise, the unproven goals may be used to uncover hidden assumptions or attacks. The NSPK protocol is analyzed for the proof of concept.

Zhang *et al.* in [350] described an extension of Paulson's inductive protocol verification approach. With this extension, *liveness* properties can be verified keeping no change of the system model underlying the original inductive approach. The newly developed extension is used in [346] to prove the correctness (liveness) of the secure routing protocol (SRP).

Zhong *et al.* in [358], based on Paulson's inductive approach and using Isabelle/HOL theorem prover, created a GM (Garay and MacKenzie) protocol model that can contain infinitely many signatories and contract texts signing simultaneously. The protocol cryptographic primitive and arithmetic including dishonest signers and outside intruder models are formalized. They found several serious problems with fairness of the existing revised protocol. A comparative study of GM and BW (Baum and Waidner) contract signing protocols is made and based on this study a new revised version of GM protocol is proposed.

2.8 Formal Methods for Additional Support of Security Protocols Development

The aforementioned formal methods described so far support the formal specification and verification for security protocols. In this section, further useful formal approaches for protocol synthesis and repairing are described.

2.8.1 Approaches of Protocol Design and Synthesis

Gong and Syverson in [188] have developed a methodology to facilitate the design and analysis of security protocols. They defined a notion for a fail-stop protocol,

which should then be verified. The authors have come out with a method that produces protocols that are fail-stop and whenever a protocol is not, their method suggests changes to turn into one that is. They have claimed, if a protocol is fail-stop protocol, it is guaranteed to satisfy secrecy assumptions. To validate this claim, BAN logic has been used.

Perrig and Song in [291] have developed Automatic Protocol Generation (APG) system for automatically generating security protocols. APG automatically generates protocols, step by step, taking into account the desired security requirements (authentication and secrecy). APG generates a collection of candidate security protocols that satisfy the specified security requirements. In the final step, APG applies *Athena* tool for analyzing the candidate protocols, discards the flawed protocols, and outputs the correct protocols that satisfy the desired security requirements. The search space of this approach is of order 10^{12} according to the authors.

Clark and Jacob in [122] presented a framework for automatic designing and synthesis of security protocols based on forward evolutionary search techniques. They used a sort of *genetic algorithm* (GA) as a heuristic search technique. BAN logic style was used to specify the design goals, from which a collection of abstract protocols were generated. The generated protocols were analyzed in relation to their BAN logic specifications. Furthermore, the tool is able to derive sets of assumptions (specification synthesis), which is usually done informally by designers of real protocols.

Saidi in [313] developed a method for automatically generating security protocols from their logical specification. This method is based on the well-known BAN logic [95], for describing protocol goals, and extended by protocol derivation rules that allow the derivation of messages from logical statements. A prototype of the method is implemented using OCaml language as it is claimed by the author.

Guttman in [194] developed a method for designing security protocols based on the authentication tests method [193]. This is illustrated by creating a ATSPECT (*Authentication Test-based Secure Protocol for Electronic Commerce Transactions*) design process. The design process is based on authentication tests and the verification method is based on the strand space theory. The steps of the design process are: (1) precise formulation of the protocol goals, (2) selecting an authentication test pattern to achieve each goal, design sub-protocols that achieve each goal, verify that the sub-protocols achieve the individual goals and ensure that the sub-protocols are independent, and (3) construction of a single protocol by combining the sub-protocols and justifying its correctness. This method is illustrated by generating a SET-like three-party protocol.

Foley and Zhou in [359, 175] proposed an automatic security protocol generator called ASPB (Automatic Synthesis Protocol Builder). The generator combines and automates the manual synthesis rules from the logic (BSW logic) proposed in [96] with Guttman manual design process (called *Authentication Tests*) [193]. The synthesis rules of the BSW logic are used to guide an automatic backwards search for a sub-protocol from a single goal. The tool combines the generated sub-protocols into one candidate protocol that matches the given goals. A time comparison between

ASPB and APG [291] based on the examples of protocols generated, was also given by the authors.

Chen in [198] developed a heuristic search framework that extended the Clark-Jacob approach [122] for synthesizing symmetric key BAN protocols to allow public key and hybrid cryptographic schemes and using SVO logic [333] - more realistic belief logic - to specify security requirements.

Xue et al. [351] used an artificial immune algorithm (AIA) to automate the design of security protocols. the cord calculus is applied as a specification formalism of security protocols. This approach is limited for simple goals.

Bela [183] developed a method for generating security protocols by composing them from other protocols. This method is called *Preconditions and effects (PE) composition*. For automating this method, an enriched protocol model that contains enough information to compose the protocol preconditions and effects and a verification approach of the correctness of the final composed protocol, are needed. Preconditions denote the set of properties that must be satisfied for the protocol to be executed, while the effects denote the set of properties resulting from the protocol execution. By composing preconditions and effects (i.e. PE composition), a new protocol sequence that ensures the satisfaction of the protocol preconditions and the propagation of generated information through effects, is generated. The protocol sequence generated by the PE composition must be correct, in the sense that it must maintain the security properties of the original protocols. The independence of the involved protocols is verified using the method in [61]. Protocol independence ensures that the intruder can not replay messages from one protocol to another to construct new attacks while running the protocols in the same environment. This property also ensures the correctness of the composed protocol.

2.8.2 Approaches of Protocol Repairing

The verification of security protocols has received a lot of attention from the formal methods community, yielding two main verification approaches: (i) *sate exploration* (model checking) methods, e.g. FDR, AVISPA, etc; and (ii) *theorem proving*, e.g. Isabelle, SPASS, etc . Furthermore, the complementary principles and guidelines of Abadi and Needham [20] of security protocol design aim to make security protocols simple and, hopefully, correct. These principles try to avoid common features (of protocols), which are hard to analyze.

The automated repairing (patching) of faulty security protocols is related to verification but less explored. To this end, Lopez et al. [297] proposed a method for patching security protocols that are susceptible to an interleaving-replay attack. This method is based on Abadi and Needham's principles of the prudent engineering practice for cryptographic protocols. Additionally, Lopez et al. in [214] developed SHRIMP, a tool which relies on existing verification tools, capable of automatically repairing faulty security protocols. SHRIMP analyzes the protocol and an attack to this protocol in order to pinpoint faulty steps in the protocol and then suggests appropriate changes to fix them. This yields an improved version of the protocol

that should be analyzed and possibly repaired again until no further flaws can be detected.

In this approach, Lopez et al. introduced a collection of rules, they called patch methods, each of which is able to deal with a class of faults. To identify and patch a protocol flaw, each rule makes use of Abadi and Needham's design principles, which Lopez et al. translated into formal requirements on sets of protocol steps. SHRIMP deals with the full class of replay attacks proposed by Syverson [331] (the only exception being the type flaw subclass). SHRIMP has been tested on 36 protocols, 21 out of which were borrowed from the Clark and Jacob library [124], obtaining a repair rate of 90%.

Another work targeting the automated repairing of security protocols is done by R. Choo [115]. He uses an adversary model from the computational complexity paradigm (i.e., Bellare and Rogaway adversarial model [67]) and an automated tool (i.e., Simple Homomorphism Verification Tool (SHVT) [279]) from the state-space exploration paradigm. The model checker (SHVT) is used to perform state-space analysis on protocol model, which is encoded using Asynchronous Product Automata (APA). If the protocol is faulty, Choo's approach automatically repairs it.

2.9 Discussion and Conclusion

We have described several research works of formal analysis of security protocols. However, there exists a huge body of research of formal methods which support analyzing security protocols. This survey highlights some of the important research works in this area. In next section, we discuss the mentioned researches in different aspects.

2.9.1 General Discussion

Here, we present a general discussion of the security verification methods introduced in the previous sections. As mentioned in this chapter and the previous chapter, there are three classes of formal methods of security protocol analysis, viz.: (1) *belief logics*, (2) *state exploration methods (model checking)*, and (3) *theorem proving*.

Using belief logics approach has the advantage of; it is easy to construct protocol models and it produces simple and short proofs and it treats the timestamps in an abstract way. In the other hand, the disadvantages of this approach are: (1) no automation tools for producing proofs, this cause a problem in case of complex protocol analysis, (2) it is limited to verify the authenticity property, (3) it assumes that all agents are honest.

In general there are two main branches of formal verification methods; model checking and theorem proving. Table 2.1 shows the most important classical features of both approaches.

Usually, the model checking problem $M \models \Phi$; involves the construction of an abstract model M, in the form of explicit transition graphs or BDDS (or other sim-

Feature	Model Checking	Theorem Proving
Automation	Fully automatic	Usually requires extensive user interaction
Specification language	Domain-specific ($M \models \Phi$)	Often HOL or FOL formula ($\models \Psi_M \rightarrow \Phi$)
Domain size	Finite	Infinite
State space explosion	Inherent limitation	No state space explosion
Number of agents and sessions	Limited number	Unlimited

Table 2.1: Advantages and disadvantages

ilar symbolic representation), and the construction of specification formulas Φ (or property specification), in the form of various temporal logics. The model checking verification problem establishes that the model semantically entails the specification of the property. Since model checking tools are made as automatic as possible, *completeness* and *termination guarantee* of model checking have been an important challenge of the research community for a long time [301, 172, 299, 300]. Completeness and termination guarantee of model checking enable the tool to guarantee the correctness of the model with respect to the specified property, or otherwise produce a counterexample. This decidability problem limits the scope of application of model checking to finite models. Also, the *efficiency* feature, which is another important feature, limits the property specification to temporal logics or small finite automata. The advantages of model checking approaches include: high-level specification languages, full automation, good for finding attacks and ability of verifying different security properties. The frequently cited drawback of this approach is the state space explosion problem. Another drawback is due to that only a candidate model of the protocol is analyzed. This may omit some other features of the protocol.

In the case of theorem proving, the higher order logic (HOL) or the first oder logic (FOL) is used as an input language for both the model and the property specification. The formula $\models \Psi_M \rightarrow \Phi$ can be read as "if property Ψ holds before M starts, Φ holds after the execution of M". The expressiveness of theorem proving input languages makes it impossible to completely automate decision procedures from the start. Therefore the user guidance is required. The distinct advantages of theorem proving over model checking is in the large size of the systems they can be applied to and their inductive reasoning. However, they have also several drawbacks. The most cited drawback of theorem proving is that they require a great deal of user expertise and effort [42]. Furthermore, the generated proofs can be large and difficult to understand [219].

However, the focus of the thesis is on the application of model checking for analyzing security protocols. In the next section, the verification scope of the major

model checking approaches, which have been described above in section §2.7.3, is discussed.

2.9.2 Verification Scope of Model Checking Tools

In this section, we discuss the verification scope of the major model checking approaches for security protocols (the general-purpose model checking tools (e.g. Spin) are not discussed here). We mean by the verification scope of a tool; the anaylsis capabilities a tool can support. This includes: supporting of time features, security properties the tool able to analyze, state space, specification language, and so on. Table 2.2 concludes those features of the most important security model checking approaches that have been discussed previously.

In general, most of the studied approaches have targeted only two security properties namely secrecy and authenticity. Also the majority of approaches does not support analysis of an unbounded number of runs of a protocol. Only ProVerif, Scyther and TA4SP can support analysis of an unbounded number of runs for a class of protocols.

Feature	Casper/FDR	Avispa toolset	Scyther	ProVerif
Security properties	secrecy and authenticity	secrecy and authenticity	secrecy and authenticity	secrecy and authenticity
Timed features	Time stamps	not supported	not supported	not supported
Number of agents and runs	limited	limited	unlimited number of runs	unlimited number of runs
Specification language	Casper	HLPSL	SPDL	Horn clauses or π-calculus
False attacks	no	TA4SP	no	yes
Documentation	sufficient	rich	lack of documentations	quite enough

Table 2.2: Verification Scope

Since the modeling is the time-consuming and error prone phase, the simplicity and expressiveness of the specification language play a great role here. Modeling security protocols using ProVerif is more complex than using the other tools due to the complexity of its specification languages. However, Avispa has a common input language (HLPSL) for the four back-ends. This allows the user to use different tools based on a single protocol model.

Producing an attack trace is one of the most important advantages of model checking tools. Attack traces provide a feedback to the user and they can help the user to fix the security flaw. All of Casper/FDR, Avispa toolkit, ProVerif and Scyther generate an attack trace, except the TA4SP back-end of Avispa does not generate an attack trace. However, it is possible that ProVerif and TA4SP show a false attack because they both use approximation. The problem is that TA4SP does not generate an attack trace, which makes it impossible to determine if the attack is true or false.

All the above tools provide support of various security properties, various forms of agreement [247] and synchronization [141]. ProVerif provides also support for equivalence-based properties. Among the presented tools in this section, Casper/FDR can analyze timed properties (time stamps).

Algebraic properties like Exclusive-OR (XOR) and Diffie-Hellman exponentiation are often employed in security protocols. It is important that security protocol verification tools support verifying protocols under some equational theories (like XOR). In [139] a list of algebraic properties of cryptographic operators with examples of protocols and attacks using these properties. Casper/FDR represents XOR by the operator "(+)" (e.g. $m\,(+)\,m'$) and defines a separate section (called "`#Equivalences`") for defining algebraic equivalences. The OFMC and CL-Atse tools of AVISPA also support specification and verification of XOR and Exponentiation properties. The ProVerif tool supports only the commutativity of the exponentiation. However, Küsters and Truderung extended ProVerif tool allowing it to deal with XOR and Diffie-Hellman algebraic operations. They proposed two new tools *XOR-ProVerif* [227] and *DH-ProVerif* [226] which automatically transform a protocol specification using XOR and Diffie-Hellman properties into a compatible language (i.e. Horn clauses) with ProVerif.

An advantage of AVISPA tool is that, its website[8] provides a rich documentation including: installation manuals for each of the back-ends, a general user manual, a beginners guide to HLPSL language, and a wide range library of examples for existing and user-contributed protocols. Also for Casper/FDR there is a sufficient documentation given in [310][9] [10]. ProVerif website[11] provides enough documentation to get familiar with the tool too. These include user and upgrade manuals, some examples for both input languages (Horn clauses and π-calculus) and other related publications. Scyther website[12] provides a short installation description and also some exercises for students. Furthermore, a shot user manual describes the main aspects of the specification language is included within the distribution files. The rich documentation of AVISPA makes it easy for beginners to learn the tool, specially from the examples library.

The discussion in this section is based on the related documentation of the rel-

[8] AVISPA Project: http://www.avispa-project.org
[9] See the book examples here: http://www.cs.rhul.ac.uk/books/secprot/
[10] Casper/FDR website: http://www.cs.ox.ac.uk/people/gavin.lowe/Security/Casper/
[11] ProVerif website: http://www.proverif.ens.fr/
[12] Scyther: http://people.inf.ethz.ch/cremersc/scyther/

evant tools. Several research works were dedicated to the comparison of several tools. For comparison between NRL and FDR model checker see [258] by the means of analysis of the Needham-Schroeder public key protocol. Cremers et al. in [144] presented a performance study of several tools (including Casper/FDR, AVISPA, ProVerif and Scyther) based on the study of state space exploration of these tools. Lafourcade et al. [229] compared OFMC and CL-Atse back-ends of AVISPA toolkit and ProVerif tool by the means of dealing with XOR and Diffie-Hellman properties. Furthermore, Dalal et al. in [148] studied several protocols using Scyther and ProVerif verification tools.

2.9.3 Current Research

In the earlier sections, we have mentioned several limitations and needs to strongly demand a new development and extensions of formal methods for verification of security systems. The new applications and systems and evolution of networks lead to situations where the used "Dolev-Yao" attacker model is not suitable for reasoning about new requirements. For example, availability requirements (e.g. DoS-resistant) that require quantitative information to estimate the available amount of resources to the attacker and the other agents. These information are not available in the Dolev-Yao model, despite the strong assumptions of the Dolev-Yao model (complete control of the network) which means by definition that it can perform DoS attack. A research line focuses on the development of new intruder models and new tools which can deal with the new systems and support quantitative verification. As examples of research works in this research line, we mention [36, 340].

Another research line concerns verification of trust and security service-oriented architectures such as web-services. The AVANTSSAR project [335] is the state-of-the-art foundation of formal techniques in this research area. This project is a follow-up of AVISPA project. One significant outcome of this project is the development of a modeling language (called *ASLan/ASLan++*) for specifying trust and security aspects of service-oriented systems. An integration of the proposed formal techniques into model checking tools is also provided. Furthermore, in this extend, we can find TulaFale [70], a scripting language for specifying SOAP security protocols. TulaFale specification can be automatically compiled to π-calculus and then verified using ProVerif tool.

A great challenge for formal methods for security analysis is the variability of protocol participants as well as the number of protocol sessions. Normally, the format and size of protocol messages is not simple. And the number of protocol participants and protocol sessions can be infinite. Distributed systems like sensor networks could be a good example. Sensor nodes can join and leave the network and change their roles dynamically. Also different natures of participants (smart phones, laptops, ...). This will impact the security protocols and the security verification methods.

However, new applications come out, new threats will also appear. So, new security requirements are required depending on the application. In the next era, we

expect that cloud computing security will be the pride challenge. Cloud computing is a distributed system depending heavily on services. This will rise a big challenge in security requirements and analysis methods of those requirements.

2.9.4 Conclusion

In this chapter, we reviewed fundamental knowledge on formal methods of security protocols and provided a survey of numerous significant works and tools. Formal methods like *model checking techniques* and *theorem proving* have been intensively used for analysis and verification of security protocols, however, formal methods are not restricted only for analysis and verification purposes but they have also been used for supporting other aspects of protocol development process such as protocol synthesis and repairing.

Security requirements (or goals) such as confidentiality, integrity and availability were introduced. Threat models (i.e. disclosure, deception, disruption or usurpation) against these goals were also briefly described. Furthermore, we briefly described security mechanisms that aim to assure and protect security goals. These include three main efforts namely *protection*, *detection* and *recovery*. Then, a description of the important types of attacks was given.

Formal techniques of security protocols were classified into three categories: belief logics, state space exploration (model checking) and theorem proving. This chapter have focused in particular on model checking techniques and tools for analysis of security protocols. However, we described also several theorem proving techniques and also several techniques for protocol synthesis and repairing. Each of these techniques has its pros and cons. The significant advantage of theorem proving over model checking techniques is in the large size of systems that it can handle. On the other hand, model checking techniques have the advantages of high-level specification languages and the full automation of verification.

Additionally, we discussed several model checking tools that have been used widely in the verification of a wide range of security protocols viz, AVISPA toolkit, Casper/FDR, Scyther and ProVerif. These tools can reason about two main properties namely secrecy and authenticity. Some of these tools also lack support of some algebraic properties which are often used in security protocols such as XOR and Diffie-Hellman properties.

Finally, a discussion of some current research topics in the area of formal methods of analysis and verification of security protocols was made. The large use of distributed systems like sensor networks and cloud computing in today computation systems rise new security challenges. Thus, new methods for verifying the new solutions are required.

Chapter 3

Case Study: Handover Authentication

Contents

3.1 Overview

Mobility is a key feature of wireless communication systems, which should be offered to users even if the networks are managed by different operators. Whenever a mobile node (MN) changes its attachment point from one domain to another one, a new authentication session should be established. Each time a MN visits a new domain it has to establish a new mutual authentication, this can cause long delays. In order to shorten such delays and support fast mobility, an inter-domain re-authentication is required between the visited domains.

Protocol for Carrying Authentication for Network Access (PANA) [176] is a link-layer transport protocol used to authenticate users before granting network access. PANA is an IP based protocol that enables the client to interact with a back-end AAA server, deployed in the network provider's domain. It enables the client to authenticate against the AAA server without using link-layer specific mechanisms or knowing the specific AAA protocol. PANA uses EAP to transfer authentication data. Thus, any authentication mechanism on top of EAP can be used.

Without extensions to PANA, whenever a mobile node (MN)[1] changes its attachment point from one domain to another domain, a new authentication session

[1] In this chapter, we use both mobile node (MN) and PANA client (PaC) interchangeably.

should be established. In some cases, it is necessary to execute the EAP exchange from scratch while in other cases it might be possible to benefit from a state stored in the visited domain's AAA server. Establishing EAP authentication from scratch each time a MN visits a new domain can considerably degrade performance. In order to enhance IP handover in mobile networks, several works were proposed. Here, we focus on two of these works [82, 35], which both of them have utilized the Context Transfer Protocol (CTP) [246] for PANA.

The protocol proposed in [82] defines a PANA context and a triggering mechanism for transferring this context between the old and new points of attachment. This protocol considers only the *intra-domain*[2] scenario. This protocol enhances intra-domain handover by transferring the already established authentication context from the current PAA (cPAA) to the new PAA (nPAA). For the protection of CTP messages, the PAAs must share IPsec security associations.

The work in [35] proposed a fast re-authentication mechanism for *inter-domain*[3] handover based on PANA context transfer. The transfer takes place between PAAs and thus the domains must trust each other (a service level agreement (SLA) is shared between them).

3.2 Terminology

The following terms, notations and abbreviations are used in this chapter for describing the above protocols and also used in HLPSL models of the protocols. Most of these terms are defined in [82, 35] and the other relevant documents (e.g., [176, 246]).

Protocol for Carrying Network Authentication for Network Access (PANA): PANA is a link-layer protocol runs between a client (PaC) and a server (PAA) in order to provide authentication and authorization for network access service. PANA's messaging consists of series of requests and answers, some of which may be initiated by either PaC or PAA. The main purpose of PANA is establishing EAP session between PaC and PAA.

PANA Client (`PaC`): PaC is a mobile node (MN) using a PANA protocol to authenticate itself to the network. It is responsible for providing the credentials to authorize itself for network access.

PANA Authentication Agent (`PAA`): PAA is the server side of the PANA protocol. A PAA is responsible for verifying the credentials provided by a PaC and authorizing network access to the PaC.

Current/Previous PANA Authentication Agent (`cPAA`): cPAA is the current PaC's

[2]Here the PAAs involved in the context transfer belong to the same administrative domain.
[3]cPAA and nPAA belong to different domains.

(or default) PAA prior to handover.

New PANA Authentication Agent (nPAA): nPAA is the PAA in charge of the subnet (or new visited domain) to which the PaC is attached after handover.

New Access Router (nAR): The router to which the PaC (or MN) attached after handover.

Current/Previous Access Router (cAR): The router to which the PaC (or MN) was attached before handover.

Home Domain (HD): HD is the domain, where the mobile node (MN) is subscribed.

New Visited Domain (nVD): nVD is the domain to which the PaC (or MN) attaches after handover.

Current Visited Domain (cVD): cVD the domain to which the PaC (or MN) was attached before the handover.

New AAA server (nAAA): nAAA is the AAA server of the nVD to which the PaC attaches after handover.

Current/Previous AAA server (cAAA): cAAA is the AAA server of the cVD to which the PaC was attached prior to handover.

Context Transfer Protocol (CTP): CTP [246] has been developed by the Seamoby Working Group at the IETF to provide seamless mobility. The basic idea of CTP is to carry the context information between Access Routers (ARs).

Context Transfer DATA (CTD): This message embeds the context to be transferred.

Context Transfer DATA Reply (CTDR): It is a reply to CTD.

PANA Client Initiation (PCI): When a PaC initiates a PANA session, it sends a PCI message to the PAA.

PANA Authentication Request (PAR): When a PAA receives a PCI message, it must respond to the PaC with a PAR message.

PANA Authentication Answer (PAN): A PaC sends a PANA-Auth-Answer message, when it receives a PAR message from a PAA.

$\{M\}_K$: Message M encrypted with key K.

`X.Y`: HLPSL message composed by concatenating (".") X and Y.

In this case study, we use the AVISPA toolkit in order to analyze the protocol reported in [35]. In next section §3.3, we introduce the AVISPA toolkit and in latter sections we describe the analyzed protocol and discuss the results.

3.3 AVISPA Toolkit

In this case study we use the *AVISPA* verification toolkit. *AVISPA* is a push-button tool for the **A**utomated **V**alidation of **I**nternet **S**ecurity-sensitive **P**rotocols and **A**pplications. It provides a modular and expressive high-level formal language HLPSL [112] for specifying protocols and their security properties and integrates different back-ends that implement a variety of state-of-the-art automatic analysis techniques. The main advantages of *AVISPA* are:

- It provides an expressive and modular specification language that allows several behaviors to be defined from a single role definition.
- AVISPA framework integrates four back-ends: SATMC, OFMC, Cl-AtSe and TA4SP. Each back-end uses different model checking technique. The protocol is specified using HLPSL language and translated by AVISPA into an intermediate format (IF) which is the input of the four back-ends of AVISPA toolkit. This Allows analyzing each model using four different tools.
- In general AVISPA has a good performance.

The current version of AVISPA integrates four back-ends: SATMC, OFMC, Cl-AtSe and TA4SP. Following a brief description of these four techniques (see Fig. 3.1).

1. *On-the-Fly Model-Checker (OFMC)* [57] is a symbolic model checker that combines two ideas based on lazy, demand driven search. The use of lazy data types is an efficient way to model check on-the-fly security protocols with large, or even infinite state spaces. OFMC also integrates a number symbolic, constraint-based techniques and optimizations for modeling a lazy Dolev-Yao intruder whose actions are generated in a demand-driven way. OFMC considers both typed and untyped protocol models.
2. *SAT-based Model-Checker (SATMC)* [48] considers the typed protocol model and performs both protocol falsification and bounded session verification by reducing the input problem to a sequence of invocations to state-of-the-art SAT solver. Thus, given as input the protocol specification and associated security properties, SATMC builds a sequence of propositional formulae which is then fed to a SAT solver and any model found is translated back into an attack.
3. *Constraint Logic based Attack Searcher (Cl-AtSe)* [114] applies constraint solving to perform both protocol falsification and verification for bounded number

of sessions. Several properties of the XOR operator can be handled, and also it is open to extensions for handling other algebraic properties of cryptographic operators. CL-AtSe performs several kinds of optimizations to reduce, and often eliminate, redundancies or useless branches in the protocol symbolic execution. For instance,the *lazy intruder technique* represents terms symbolically to avoid explicitly enumerating the possible messages the Dolev-Yao intruder can generate.

4. *Tree Automata based on Automatic Approximations for the Analysis of Security Protocols (TA4SP)* [77] performs unbounded protocol verification by approximating the intruder knowledge by using rewriting and regular tree languages. TA4SP only checks secrecy property.

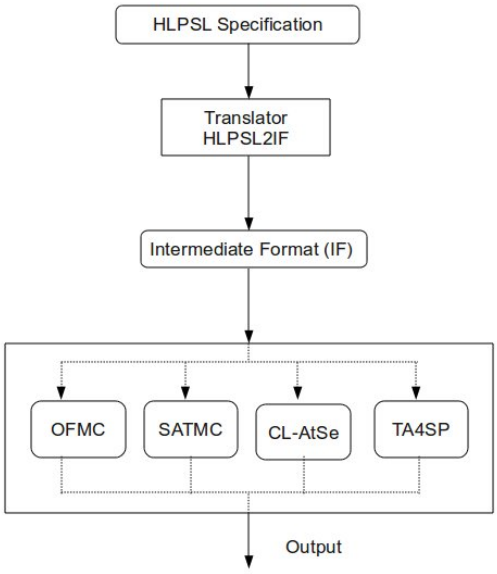


Figure 3.1: AVISPA Architecture

A graphical interface SPAN [186] is also provided by AVISPA. The role of SPAN is to symbolically execute a HLPSL protocol specification so as to have a better understanding of the specification, check that it is executable and that it corresponds to what is expected.

In AVISPA there are two kinds of roles; basic roles and composed roles. A basic role describes a protocol participant initial knowledge, its initial state and a set of transitions that describes the behavior of the participant. One or more basic roles can be instantiated to form a composed roles, which describe sessions of the protocol.

3.4 Verification of The Inter-Domain CTP for PANA

In this section, we will present the analysis and verification of the re-authentication framework for inter-domain handover -(we call it *inter-domain CTP for PANA*)- that was proposed in [35].

This framework provides an automated authentication mechanism to avoid the establishment of a new authentication process whenever a user (mobile node) visits a new domain (nVD). The already existed authentication credentials, of the current session in the current visited domain (cVD), are re-established in advance and utilized for the authentication process in a new visited domain. This inter-domain re-authentication requires that a service level agreement (SLA) among the operators of the domains exists. The PANA CTP is used to transfer the authentication contexts from the current visited domain (cVD) to the new visited domain (nVD) without involving the user's (PaC or MN) home domain (HD).

To guarantee an authenticated access of network resources, the common framework incorporates an AAA infrastructure. The proposed scenario in [35] considered three domains: the home domain, the current visited domain and the new visited domain, each with its own AAA server.

3.4.1 Description of The Protocol

The protocol consists of 12 messages. Following, a description of the protocol messages using the Bob-Alice notation. The protocol messages are coded in HLPSL language.

The protocol starts when PCI (PANA-Client-Initiation) is sent by the Mobile Node (MN) or the PANA Client (PaC) to the current PANA Authentication Agent (cPAA). The PCI contains the new ID of the PaC (NewPaC_ID) and the ID of the new PANA authentication agent (NPAA_ID) as the following:

PaC (MN) $\rightarrow$ *CPAA: NewPaC_ID.NPAA_ID*

Before transferring the context the cPAA checks if a service level agreement (SLA) between the current visited domain (cVD) and the new visited domain (nVD) exists. This is checked by sending a Domain-Trust-Request (DTR) containing the nPAA_ID to the cAAA. To confirm the cAAA sends the nAAA's public key (`NAAA_PK`) in a Domain-Trusted-Answer. The `NAAA_PK` is used later to check the integrity of the messages from the nPAA:

CPAA $\rightarrow$ *CAAA: CPAA_ID.NPAA_ID*
CPAA $\leftarrow$ *CAAA: CPAA_ID.NAAA_PK*

After the SLA is checked the cPAA forwards the PCI to the nPAA. The PCI includes the nPAA's own ID and the new ID of PaC. The nPAA interprets the other ID as the ID of the PaC requesting inter-domain PANA CXTP from the

sender of the PCI (the cPAA) to the nPAA. This message is written as the following:

CPAA → NPAA: NewPaC_ID.NPAA_ID

The nPAA will only accept transferred context if an adequate SLA exists. It sends a DTR to the nAAA and gets as confirmation a DTA containing cAAA's public key (`CAAA_PK`) as the following:

NPAA → NAAA: NPAA_ID.CPAA_ID
NPAA ← NAAA: NPAA_ID.CAAA_PK

Then the nPAA starts PANA authentication with the PaC by sending a PANA Authentication Request (PAR) to the PaC containing a fresh nonce (`NnPAA`) and the start bit set (s-bit) as following:

NPAA → PaC: S_BIT.New_ns_id.NnPAA

The PaC replies with a PANA Authentication Answer (PAN) signed by the `PANA_Auth_Key` containing a fresh nonce (`NPaC`) and the start bit set (s-bit) as the following:

PaC → NPAA: [S_BIT.New_ns_id.NPaC]_PANA_Auth_Key

The nPAA starts the context transfer by sending a Context-Transfer-Request (CTR) to the cPAA containing the last PAR message as well as the nPAA identity (`NPAA_ID`) and public key (`NPAA_PK`) both signed by the nAAA secret key (`NAAA_SK`) as the following:

NPAA → CPAA: PAR.[New_ns_id.NnPAA.NPAA_ID.NPAA_PK]_NAAA_SK

The cPAA checks the integrity of the PAR message and validates the identity and public key of the nPAA. If all checks pass, the cPAA computes an intermediate AAA Key (`AAA_Key_Int`). The `AAA_Key_Int` is the PANA context used by nPAA to authenticate the PaC in the nVD. The cPAA transfers the PANA context encrypted under the nPAA's public key (`NPAA_PK`) to the nPAA into a Context-Transfer-Data message (CTD) such as the following:

CPAA → NPAA: {AAA_Key_Int}_NPAA_PK

The nPAA sends a PAR to the PaC signed by the `New_PANA_Auth_Key` with the complete bit set (c-bit) containing the new key id (`New_Key_id`) and algorithm ID (`Algo_id`). As the Following:

NPAA → PaC: [P_BIT.New_ns_id.New_Key_id.Algo_id]_New_PANA_Auth_Key

The PaC completes the protocol run by sending a PAN signed by the `New_PANA_Auth_Key` with the complete bit set (c-bit) containing the `New_Key_id`. As the following:

PaC → NPAA: [New_ns_id.New_key_id]_New_PANA_Auth_Key

```
PaC, CPAA, NPAA, CAAA, NAAA : agent
PANA_Auth_Key, AAA_Key_Int, New_PANA_Auth_Key, New_AAA_Key, NAAA_SK
: symmetric_key
NPAA_PK, CAAA_PK, NAAA_PK : public_key
NewPaC_ID, NPAA_ID, CPAA_ID, NPaC, NnPAA, New_ns_id, Algo_id,
New_Key_id :  text
Snd, Rcv :  channel(dy)
npaa_pac_panakey, sec_panakey, aaaint :  protocol_id
KeyGen :  hash_func
```

Figure 3.2: Significant HLPSL Declarations

3.4.1.1 Significant HLPSL Declarations of the Model

The specification language of AVISPA, HLPSL is a role-based language. In this model, five roles are defined, each of them represent a principal (or agent) in the protocol run. Figure 3.2 shows the HLPSL declarations of the agents and the other parameters of the protocol Followed by a brief explanation of the used syntax.

PaC, CPAA, NPAA, CAAA, NAAA : agent

The values of the type **agent** represent the names of the protocol principals. The above line defines the principals participating in the protocol session. `PaC` is the PANA-Client or the Mobile Node (MN), `CPAA` is the current PANA-Authentication-Agent which the PaC is authenticated with, `NPAA` is the new PANA-Authentication-Agent which the PaC wants to communicate with in the new visited domain (nVD), `CAAA` is the AAA server in the current visited domain (cVD), and `NAAA` is the AAA server in the nVD.

PANA_Auth_Key, AAA_Key_Int, New_PANA_Auth_Key, New_AAA_Key, NAAA_SK : symmetric_key

Variables of **symmetric_key** type represent keys for symmetric encryption. `PANA_Auth_Key` is a PANA key derived by PaC and cPAA in the cVD for the current session, `New_PANA_Auth_Key` is a PANA key derived by nPAA and PaC in

nVD for the new session, `AAA_Key_Int` is an AAA key derived by cPAA and sent to nPAA which uses it as a PANA context to authenticate PaC, `New_AAA_Key` is an AAA key derived by nPAA for the new session, and `NAAA_SK` is a secret key of the AAA server in the new visited domain (nVD).

NPAA_PK, CAAA_PK, NAAA_PK : public_key

Values of variables of **public_key** type represent agents' public keys for asymmetric cryptography. The line above defines the nPAA public key (`NPAA_PK`), the AAA server in the cVD (`CAAA_PK`) and the AAA server in the nVD (`NAAA_PK`).

NewPaC_ID, NPAA_ID, CPAA_ID, NPaC, NnPAA,New_ns_id, Algo_id,
New_Key_id : text

The variables of type **text** are often used as nonces. `NewPac_ID` is the new id of the MN or PaC, `NPAA_ID` is the nPAA's id, `CPAA_ID` is the cPAA's id, `NpaC` is a nonce generated by PaC, `NnPAA` is a nonce generated by nPAA, `New_ns_id` is the numeric session id, `Algo_id` is an id of an integrity algorithm determined by nPAA to specify the pseudo-random function (PRF) to be used for the derivation of the New_PANA_Auth_Key, and *New_Key_id* is a computed value used to check the integrity of the PANA key.

npaa_pac_panakey, sec_panakey, aaaint : protocol_id

Values of **protocol_id** type are labels used to identify the goals specified in the goal section. The goals are the properties to be checked by the tool.

Snd, Rcv : channel(dy)

The variables of type **channel** are variables over which the communication takes place. The attribute (dy) specifies the intruder model to be used for the communication over the channel. `Snd` and `Rcv` are the send and receive channels, respectively in the Dolev-Yao intruder model.

KeyGen : hash_func

The hash function used to derive the keys.

3.4.1.2 HLPSL Syntax of the Model

The parameters employed in the previous section represent the basic common parameters of the HLPSL model of the protocol. Due to the length and safe space, we present some parts of the HLPSL specification of the model.

The following list (List 3.1) shows the parameters and local variables sections of

the basic role `panaClient` which simulates the behavior of the PANA Client (PaC) in the protocol. The parameters `PaC, CPAA, ...` that have been described in the previous section §3.4.1.1, model the pre-shared knowledge among the protocol participants. For example, the `PaC` and `nPAA` must agree about the generating algorithm of the `New_PANA_Auth_Key` key.

As a convention, all variables in HLPSL start with a capital letter and constants start with a small letter. The section `played_by` tells that `PaC` is the name of agent who plays the role `panaClient`. Also the `local` section declares the local variables of the role.

The protocol behavior played by the role is specified as a set of transitions and presented in the `transition` section of the HLPSL specification. A transition consists of a trigger and an action to be performed when the trigger event occurs. In general, each transition represents the receipt of a message and the sending of a replay message.

```
role panaClient(
        PaC, CPAA, NPAA,
        CAAA, NAAA              : agent,
        PANA_Auth_Key           : symmetric_key,
        CPAA_ID, NPAA_ID        : text,
        KeyGen                  : hash_func,
        Snd, Rcv                : channel(dy)
)
played_by PaC def=
local
        State                   : nat,
        New_Key_id              : text,
        Algo_id,
        NPaC, NnPAA,
        New_ns_id               : text,
        NewPaC_ID               : text,
        New_PANA_Auth_Key       : hash(hash(text.text.text).text.text.
            text)


const
        npaa_pac_panakey,
        sec_panakey             : protocol_id


init State := 0

transition

  1. State = 0
        /\ Rcv(start)
  =|> State' := 1
        /\ NewPaC_ID' := new()
```

```
35          /\ Snd(NewPaC_ID'.NPAA_ID)
36
37    2. State = 1
38          /\ Rcv(New_ns_id'.NnPAA')
39
40    =|> State' := 2
41          /\ NPaC' := new()
42          /\ Snd({New_ns_id'.NPaC'}_PANA_Auth_Key)
43
44    3. State = 2
45          /\ Rcv({New_ns_id.New_Key_id'.Algo_id'}_New_PANA_Auth_Key')
46
47    =|> State' := 3
48          /\ Snd({New_ns_id.New_Key_id'}_New_PANA_Auth_Key')
49          /\ witness(PaC, NPAA, npaa_pac_panakey, New_PANA_Auth_Key')
50  end role
```

Listing 3.1: HLPSL specification of the panaClient role

The other basic roles; `currentPanaAuthAgent`, `newPanaAuthAgent`, `currentAAA` and `newAAA` are defined similarly as the role `panaClient` above. Specification of the security goals is described in the next section §3.4.2.1.

Another kind of roles are the composed roles. There are two composed roles namely: `session` role and `environment` role. Instead of the `transition` section, the composed roles have the `composition` section in which the basic roles are instantiated.

The `session` role of our protocol specification is shown in List 3.2. Usually, all the channels, that are used by the basic roles, are declared in the `session` role. In this protocol model, we declared 10 channels, for each agent two channels. One for sending messages (e.g. `SPaC` which means the sending channel for the `panaClient` role). The other channel for receiving messages (e.g. `RPaC` which means the receiving channel of the `panaClient` role). The proceeding letter (S/R) before the name of the agent variable indicates the type of channel (`S` for sending and `R` for receiving). Those channels are defined in the `local` section of the `session` role that is shown in the List 3.2. The `composition` section of the `session` role instantiates one instance of each basic role, (of the basic roles: panaClient, currentPanaAuthAgent, newPanaAuthAgent, currentAAA and newAAA), which describes one whole protocol session (or run). Thus, the `session` role, in our model, assumes five basic roles together and defines for each role, what information it begins with by passing this in as arguments. The operator /\ indicates that the roles are executed in parallel.

```
1
2  role session(
3          PaC, CPAA, NPAA,
4          CAAA, NAAA                  : agent,
5          PANA_Auth_Key               : symmetric_key,
6          CPAA_ID, NPAA_ID            : text,
```

```
7           NPAA_PK, NAAA_PK,
8           CAAA_PK                 : public_key,
9           Key_id                  : text,
10          KeyGen                  : hash_func
11          %NAAA_SK                : symmetric_key
12
13  )
14
15  def=
16
17  local
18          SPaC, SnPAA,
19          ScPAA, RPaC,
20          RnPAA, RcPAA,
21          ScAAA, RcAAA,
22          SnAAA, RnAAA            : channel(dy)
23
24  composition
25          panaClient(
26          PaC, CPAA, NPAA, CAAA, NAAA, PANA_Auth_Key, CPAA_ID, NPAA_ID,
                KeyGen,SPaC, RPaC
27                  )
28      /\  newPanaAuthAgent(
29          PaC, CPAA, NPAA, CAAA, NAAA, CPAA_ID, NPAA_ID, NPAA_PK,
                NAAA_PK, CAAA_PK, KeyGen, SnPAA, RnPAA
30                  )
31      /\  currentPanaAuthAgent(
32          PaC, CPAA, NPAA, CAAA, NAAA, CPAA_ID, NPAA_ID, Key_id, KeyGen,
                ScPAA, RcPAA
33                  )
34      /\ currentAAA(
35          PaC, CPAA, NPAA, CAAA, NAAA, CPAA_ID, NPAA_ID, CAAA_PK,
                NAAA_PK, ScAAA, RcAAA
36                  )
37      /\ newAAA(
38          PaC, CPAA, NPAA, CAAA, NAAA, CPAA_ID, NPAA_ID, CAAA_PK,
                NAAA_PK, SnAAA, RnAAA
39                  )
40  end role
```

Listing 3.2: HLPSL specification of the session role

Finally, the `environment` role is the top-level role which contains the global constants and a composition of one or more sessions (protocol runs). The intruder may play some roles as a legitimate user, in one some of these sessions. Another statement in the `environment` role is the `intruder_knowledge`. Which describes the initial knowledge of the intruder. Usually, this includes the public keys, the names of agents, the intruder's private key, other keys he shares with others and the publicly known functions. In the `environment` role, the constant `i` is used to refer to the intruder. The Listing 3.3 shows the `environment` role of our protocol model.

The `composition` section defines four sessions of the protocol. The first session is the so-called *normal session* whose participating roles are only the legitimate

users of the protocol. In the other three sessions, the intruder `i` plays as `PaC`, `cPAA` and `nPAA`, respectively. For verifying the protocol, we test several scenarios which considers a combination of those sessions (more details about the model verification are described in sections §3.4.3).

```
1  role environment()
2 def=
3
4    const
5          npaa_pac_panakey,
6          sec_panakey,
7          aaaint                              : protocol_id,
8
9          pac, cpaa, npaa,
10         caaa, naaa                          : agent,
11
12         pana_auth_key_i,
13         pana_auth_key,
14         aaa_key_int_i,
15         new_pana_auth_key_i,
16         naaa_sk                             : symmetric_key,
17
18         cpaa_id, npaa_id, i_id,
19         key_id                              : text,
20
21         keygen                              : hash_func,
22
23         naaa_pk, npaa_pk, caaa_pk           : public_key
24
25
26 intruder_knowledge = {pac, cpaa, npaa, caaa, naaa, pana_auth_key_i,
       aaa_key_int_i, naaa_pk, npaa_pk, caaa_pk,
27                            new_pana_auth_key_i}
28
29 composition
30             session(pac, cpaa, npaa, caaa, naaa, pana_auth_key, cpaa_id,
                    npaa_id, npaa_pk, naaa_pk, caaa_pk,
31                  key_id, keygen)
32         /\ session(i, cpaa, npaa, caaa, naaa, pana_auth_key, cpaa_id,
                   npaa_id, npaa_pk, naaa_pk, caaa_pk,
33                  key_id, keygen)
34         /\ session(pac, i, npaa, caaa, naaa, pana_auth_key, cpaa_id,
                   npaa_id, npaa_pk, naaa_pk, caaa_pk,
35                  key_id, keygen)
36         /\ session(pac, cpaa, i, caaa, naaa, pana_auth_key, cpaa_id,
                   npaa_id, npaa_pk, naaa_pk, caaa_pk,
37                  key_id, keygen)
38
39
40 end role
```

Listing 3.3: HLPSL specification of the environment role

3.4.2 Verified Goals

Three goals are verified (as shown in Figure 3.3) namely:

- strong authentication on `New_PANA_Auth_Key` (the new derived PANA authentication key).
- secrecy of `New_PANA_Auth_Key`.
- secrecy of `AAA_Key_Int` (the AAA intermediate key).

```
goal
     authentication_on npaa_pac_panakey
     secrecy_of sec_panakey
     secrecy_of aaaint
end goal
```

Figure 3.3: Specification of properties in AVISPA

- `authentication_on npaa_pac_panakey`. This goal checks the authenticity of the new PANA authentication key between nPAA and PaC. This means nPAA authenticates PaC on the New_PANA_Auth_Key.

- `secrecy_of sec_panakey`. This goal checks the secrecy of the new PANA authentication key that means to check if there is another agent, aside from nPAA and PaC, can learn this key.

- `secrecy_of aaaint`. This goal checks the secrecy of the intermediate AAA key which should be known only to cPAA and nPAA.

3.4.2.1 HLPSL Specification of Security Goals

In HLPSL, security goals are specified by augmenting the transitions of basic roles with *goal facts* and then assigning them a meaning by describing what conditions indicate an attack. HLPSL provides useful macros for the frequently used security goals, secrecy and authenticity, hiding the internal temporal logic specification of security goals.

The *witness* and *request* events are the goal facts for specifying *authentication*. In our model, we use them to specify the authentication goal of the `New_PANA_Auth_Key` key. That means both the PaC and nPAA should agree on the value of that key. The List 3.4 shows the HLPSL specification of this goal, in particular lines 13, 30 and 36.

```
% Definition of PANA Client (PaC) Role
role panaClient(
                    ....
)

played_by PaC def=
....

transition
  .
  .
  .
  /\ witness(PaC, NPAA, npaa_pac_panakey, New_PANA_Auth_Key')

end role


% Definition of New PANA Authentication Agent (nPAA)  Role
role newPanaAuthAgent(
                  ....
)

played_by NPAA def=
....

transition
  .
  .
  .
  /\ request(NPAA, PaC, npaa_pac_panakey, New_PANA_Auth_Key)

end role


goal
        authentication_on npaa_pac_panakey
        ....
end goal
```

Listing 3.4: HLPSL specification of the authentication goal

The secrecy goal is modeled in HLPSL using `secret` goal fact (or predicate). In our model, we modeled two secrecy goals; one is the secrecy of the `New_PANA_Auth_Key` key and the second is the secrecy of the intermediate AAA key `AAA_Key_Int` key. As shown in the List 3.5, Lines 12 and 37 are the HLPSL specification of the secrecy goal of the intermediate AAA (`AAA_Key_Int`) key which must be only known to `cPAA` and `PaC` agents. And Lines 29 and 36 are the specification of the secrecy goal of the new PANA authentication (`New_PANA_Auth_Key`) key which must be only known to `nPAA` and `PaC` agents.

```
% Definition of current PANA Authentication Agent (cPAA)  Role
```

```
role currentPanaAuthAgent(
                          ...
)

played_by CPAA def=

transition
  .
  .
  .
  /\ secret(AAA_Key_Int', aaaint, {CPAA, NPAA})

end role


%definition of New PANA Authentication Agent (nPAA)  Role
role newPanaAuthAgent(
....
)

played_by NPAA def=
....

transition
  .
  .
  .
  /\ secret(New_PANA_Auth_Key, sec_panakey, {NPAA, PaC})

end role


goal
        ....
        secrecy_of sec_panakey
        secrecy_of aaaint
end goal
```

Listing 3.5: HLPSL specification of secrecy goals

3.4.3 Analysis of The Model

The analysis of the model is done through several scenarios, mainly we focus on two parallel sessions scenarios. One of these sessions is the so-called "`normal session`" which is a protocol session (or run) with only the legitimate users of the protocol (the intruder plays no role in such a session). The other session is a protocol session where the intruder plays a role of one of the legitimate protocol users (`PaC`, `cPAA` or `nPAA`). We test all the scenarios, where the first session is a normal session and the second session consists of the intruder replacing, each time, one of the legitimate users; `PaC`, `cPAA`, or `nPAA`. For example, the List 3.6 shows a scenario, where, in the second session, the legitimate user `PaC` is played by the intruder.

```
role environment()
def=
...
composition
            session(pac, cpaa, npaa, caaa, naaa, pana_auth_key, cpaa_id,
                 npaa_id, npaa_pk, naaa_pk, caaa_pk,
                 key_id, keygen)
         /\ session(i, cpaa, npaa, caaa, naaa, pana_auth_key, cpaa_id,
             npaa_id, npaa_pk, naaa_pk, caaa_pk,
                  key_id, keygen)
end role
```

Listing 3.6: Verification scenario: the intruder plays the role as PaC

Listing 3.7 shows the verification result of the above scenario, which is obtained by CL-AtSe back-end of the AVISPA toolkit. This shows that the protocol is `safe` and satisfies all its goals.

```

SUMMARY
  SAFE

DETAILS
  BOUNDED_NUMBER_OF_SESSIONS
  TYPED_MODEL

PROTOCOL
  /final_model_hash.if

GOAL
  As Specified

BACKEND
  CL-AtSe

STATISTICS
  Analysed    : 120947 states
  Reachable   : 87236 states
  Translation: 0.09 seconds
  Computation: 21.01 seconds
```

Listing 3.7: CL-AtSe result of the scenario of 2 parallel sessions: where i plays as PaC in the second session

Another scenario, that we would briefly highlight, is the two-parallel sessions. In which, the attacker plays the role of the legitimate user (`nPAA`) of the protocol, in the second session as shown in the Listing 3.8.

```
role environment()
def=
```

```
3  ...
4  composition
5              session(pac, cpaa, npaa, caaa, naaa, pana_auth_key, cpaa_id,
                   npaa_id, npaa_pk, naaa_pk, caaa_pk,
6                  key_id, keygen)
7          /\ session(pac, cpaa, i, caaa, naaa, pana_auth_key, cpaa_id,
               npaa_id, npaa_pk, naaa_pk, caaa_pk,
8                  key_id, keygen)
9  end role
```

Listing 3.8: Verification scenario: the intruder plays the role as nPAA

As in the previous scenario, the CL-AtSe back-end is used. AVISPA showed that this scenario is `unsafe` and identified a security vulnerability of this protocol. The attack is a violation of the specified authentication goal of the new PANA authentication key, as shown in Listing 3.9. The generated attack trace (see Appendix A) shows that there is a replay attack. The intruder could manage to replay some messages from one session on another and could complete a protocol run with the `nPAA` masquerading the `PaC`. Figure 3.4 depicts this attacks.

```
1
2  SUMMARY
3    UNSAFE
4
5  DETAILS
6    ATTACK_FOUND
7    TYPED_MODEL
8
9  PROTOCOL
10   /final_model_hash.if
11
12 GOAL
13   Authentication attack on (npaa,pac,npaa_pac_panakey,{AAA_Key_Int(10).
         n8(New_ns_id).n28(NPaC).n8(NnPAA)}_keygen)
14
15 BACKEND
16   CL-AtSe
17
18 STATISTICS
19   Analysed    : 823 states
20   Reachable   : 647 states
21   Translation: 0.07 seconds
22   Computation: 0.03 seconds
23 ....
```

Listing 3.9: CL-AtSe result of the scenario of 2 parallel sessions: where i plays as nPAA in the second session

Further scenarios and a conclusion of the obtained results are presented in the next section §3.4.4.

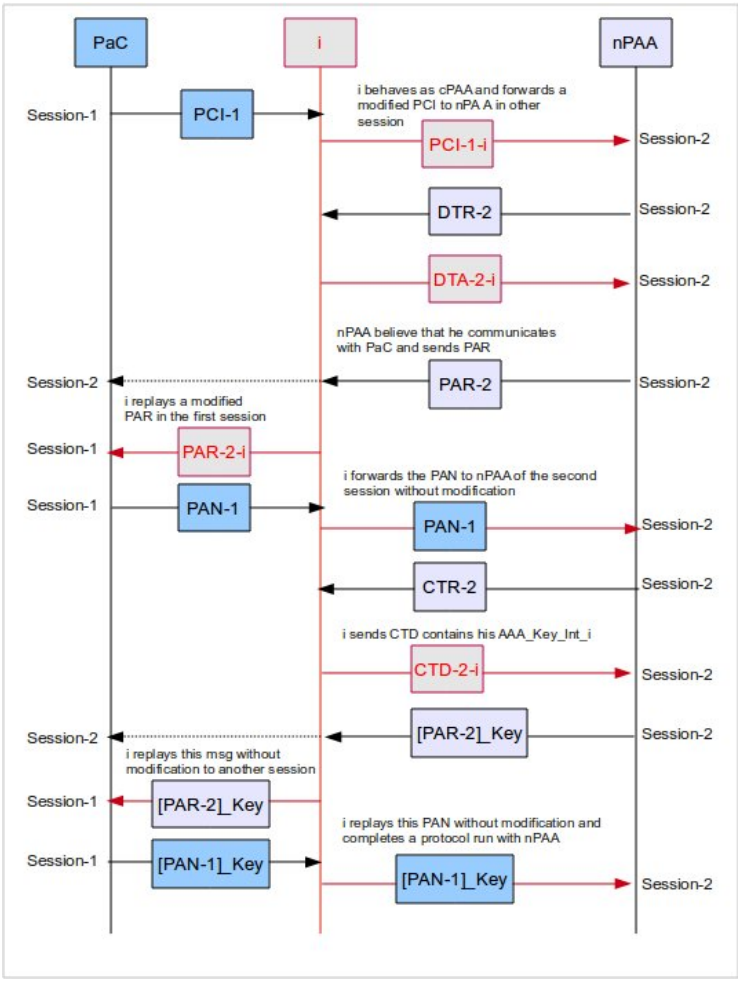


Figure 3.4: Illustration of the attack in section §3.4.3

3.4.4 Summary of The Analysis Results

In all scenarios that we have tested, except the one that has been explained previously in section §3.4.3, the protocol satisfies its security requirements and is shown to be *safe*. Table 3.1 concludes all scenarios that have been tested and the final result of the analysis (*safe*/*unsafe*) of each scenario. The second column in the table shows the participants of the protocol for each scenario. In all the mentioned scenarios, except the first two, the intruder plays a role as one of the legitimate users of the protocol (`PaC`, `cPAA`, `nPAA`, `cAAA` or `nAAA`) in the second session.

This attack shows that compromising `cPAA` can violate the protocol by pretending an unauthorized `PaC` to the `nPAA` as a authorized one. But this kind of attack, compromising PAAs, is a general risk. The protection of such common risks can be only guaranteed by operators.

Scenarios	Session parameters					Result
	PaC	cPAA	nPAA	cAAA	nAAA	
Single session	pac	cpaa	npaa	caaa	naaa	Safe
Two similar session	pac	cpaa	npaa	caaa	naaa	Safe
	pac	cpaa	npaa	caaa	naaa	
Two sessions replacing PaC with i	pac	cpaa	npaa	caaa	naaa	Safe
	i	cpaa	npaa	caaa	naaa	
Two sessions replacing cPAA with i	pac	cpaa	npaa	caaa	naaa	Safe
	pac		npaa	caaa	naaa	
Two sessions replacing nPAA with i (all goals)	pac	cpaa	npaa	caaa	naaa	*Unsafe*
	pac	cpaa	i	caaa	naaa	
Two sessions replacing nPAA with i (secrecy only)	pac	cpaa	npaa	caaa	naaa	Safe
	pac	cpaa	i	caaa	naaa	
Two sessions replacing cAAA with i	pac	cpaa	npaa	caaa	naaa	Safe
	pac	cpaa	npaa	i	naaa	
Two sessions replacing nAAA with i	pac	cpaa	npaa	caaa	naaa	Safe
	pac	cpaa	npaa	caaa	i	

Table 3.1: Summary of the Verification Results

3.5 Related Work

Model checking tools successfully have been used for the verification of security protocols in different domains such as: authentication protocols [124, 277], wireless security protocols [222], etc.

The AVISPA tool has been used to analyze several security protocols in different domains. For example, Abdelnur et al. in [24] used AVISPA to analyze the authentication mechanism of SIP (Session Initiation Protocol). An attack was identified against the SIP authentication mechanism, which can allow toll fraud or call hijacking. In [43] Armando et al. used SATMC for analyzing the SAML-based Single-Sign-On protocol for Google applications. The analysis in [43] has revealed a security flaw that allows a dishonest service provider to impersonate a user at another service provider. Furthermore, the AVISPA library [5] provides a large number of specifications of security protocols that have been analyzed by the AVISPA team. This library includes specifications of IETF, non-IETF and e-Business protocols.

3.6 Summary

This chapter shows that the proposed authentication framework in [35] is able to fulfill its security properties in almost all the tested scenarios. Table 3.1 summarizes

the results of the tested scenarios.

The presented formal verification, by using AVISPA toolkit, shows that compromising PAAs is a common risk which should be dealt with by the operators of network domains. However, further research is necessary to weaken such vulnerabilities and increase security level. This requires additional data protection solutions to be deployed by the operators of network domains.

Chapter 4

Denial-of-Service Attacks

Contents

4.1 Introduction

The price of the emphasizing of scalability and openness while designing the Internet infrastructure is very poor security. The Internet Protocol (IP) was designed to make ease attachment of hosts to networks with a little verification of the contents of IP packet header fields. This makes it possible to spoof the source IP addresses, and hence difficult to identify the source of traffic. Furthermore, IP does not provide an inherent mechanism to check whether a source address is authorized to access a service. Packets are delivered to their destination which must decide whether or not to accept and service these packets. Despite defenses such as firewalls can be added to protect servers, the main challenge for defense mechanisms is how to differentiate between legitimate requests for service and malicious ones.

In case of generating service requests by a client is cheaper than it is for a server to validate those requests, then it is difficult to protect the server from malicious requests that aim to waste the resources of the server. This makes it possible to launch a *Denial-of-Service* (DoS) attack.

Denial-of-Service (DoS) attack is an attempt to make a computer or network resources unavailable to its intended users. DoS attacks can occur in a wide variety of contexts, from operating systems [185] to network-based services [276]. On the Internet, DoS attacks aims to disrupt some legitimate activity, such as browsing Web pages, listening to an on-line radio, transferring money from your bank account, etc. Dos attacks are a serious problem on the Internet society because they are easy to generate by malicious clients but very difficult to protect by server (responder).

A Denial-of-service (DoS) attack can be either a *single-source* attack, originating at only one host, or a *multi-source* attack, where multiple hosts coordinate to flood the victim with a vast number of malicious packets.

A DOS attack can be launched in two ways [206]. One way is by sending crafted packets in order to exploit a vulnerability present in the victim. For example, the "ping-of-death" attack sends a large International Control Message Protocol (ICMP) ping packet that is fragmented into multiple datagrams to a target system, which can cause certain operating system to crash, freeze, or reboot due to buffer overflow. Another way by flooding the victim with a barrage of malicious packets that consume a key resources such as CPU time, memory, bandwidth, etc. The victim spends all of its critical resources on handling those malicious packets and cannot attend to its legitimate clients.

Of course, to generate a vast number of malicious packets, the attacker must control a very powerful machine – with a sufficiently fast processor and a lot of available network bandwidth. For the attack to be successful, it has to overload the target's resources. This means that an attacker's machine must be able to generate more traffic than a target, or its network infrastructure, can handle.

When the traffic of a DoS attack comes from multiple sources, it is called a *distributed denial of service attack* (DDoS). By using multiple attack sources, the power of a DDoS attack is amplified and the problem of defense becomes more complicated. This due to the characteristics of DDoS attack [268];

- DDoS attack uses a very large number of machines. This builds a powerful weapon, that can cause any target, regardless of how well provisioned it is, to be taken off-line. Many automated tools for DDoS attack can be found on the Internet that allow attackers to gather and engage large number of zombie machines trivially. Such tools do not require sophistication to be used and can inflict very effective damage. A large number of coordinated machines gives another advantage to an attacker.

- Some DDoS attacks use seemingly legitimate traffic. The traffic is usually so aggregated that it is very difficult to distinguish legitimate packets from attack packets. Since the attack misuses a legitimate activity, it is extremely hard to respond to the attack without also disturbing this legitimate activity.

Here we list some examples of DoS and DDoS attacks that have been registered against Internet society:

- On November 2, 1988, the Internet was infected by a *worm* program that exploited vulnerabilities in utility programs in BSD-derived version of Unix. Those vulnerabilities allowed the worm program to break into the infected machines and copy itself. Eventually, this worm spread to thousands of machines and disrupted normal activities and Internet connectivity for many days. This worm is known by *Morris worm*.

- On February 9, 2000, Yahoo, eBay, Amazon.com, E*Trade, ZDnet, Buy.com, the FBI, and several other Web sites fell victim to DDoS attacks resulting in substantial damage and inconvenience [180].

- In October 2002, there was an attack on all 13 root Domain Name System (DNS) servers, which keep important data for the whole Internet. Since DNS information is heavily cached and the attack lasted only an hour, there was no large disruption of Internet activity. This attack disabled the port of Houston, Texas, was actually directed at a South African chat room user, with the port's computers being misused for the attack [238].

- From December 2005 to January 2006, 1,500 separate IP addresses were victims of DDoS attacks, with some attacks using traffic rates as high as 10 Gb/s [220, 343].

However, despite the great advantages and services of Internet, with this internetworking infrastructure comes in the potential for the Denial of Service (DoS) attacks. DoS attack aims to reduce the availability of resources to one or more users. This is usually done by exhausting some critical limited resource (e.g. bandwidth) via packet flooding or by sending malformed packets that crash the intended resource. The number of DoS attacks is increased [204]. According to CERT, the number of reported Internet security incidents has jumped from six in 1988 to 82,094

in 2002, and to 137,529 in 2003 [106]. Due to the excessive number of security incidents, CERT has decided not to publish the number of incidents reported since 2004.

The annual surveys of the Computer Security Institute (CSI) on the prevalence and character of computer crimes based on the responses from several respondents including security analysts and Chief Security Officers (CSO) from mid-to-large firms in the U.S. [232, 303, 304]. Table 4.1 shows the percentage of the participants who were targeted by a DoS attack between 1999 and 2010.

Year	Percentage of Respondents Observing DoS Attack
1999	30%
2000	27%
2001	36%
2002	40%
2003	42%
2004	39%
2005	32%
2006	25%
2007	25%
2008	21%
2009	29%
2010	17%

Table 4.1: Percentage of CSI Cybersecurity Survey Responders Who Observed a DoS Attack During 1999-2010.

Nowadays, Bot-nets are the armies for attackers to launch DDoS attacks. Bots have been openly sold. In 2010 Symantec observed an advertisement promoting 10,000 bots for $15 [138]. And the largest bot-net observed in 2010 (called *Rustock*), had over one million bots under its control [138].

The increase of DoS attacks were met by several applications that claim DoS resilience. However, there is much attention that have been done for designing effective DoS defense mechanism, but little has been done for developing effective and realistic tests to evaluate those mechanisms. Developing such realistic tests consumes more time in investigating the test setup and design than in developing a solution itself. This is because of the lack of shared means and the complexity of DoS phenomenon. Testing DoS resilience claims requires some metrics in order to measure the service to DoS attacks.

4.2 Types of DoS Attacks

As mentioned above DoS attacks can be launched from a single source or multiple sources. Attacks that are launched from multiple sources are called Distributed DoS

attacks (DDoS), their impact on the victim is more severe. In DDoS attacks, the attacker employs a large number of zombies to send bogus packets simultaneously to crash the victim. As an example of a DDoS attack is *Tribe Flood Network 2K* (TFN2K) attack [52].

Needham [276] stated three basic types of DoS and DDoS attacks; attacks on the server, attacks on the network and attacks on the client. Attacks against the server attempt to interrupt the server to deny the legitimate requests. Attacks on the network, mainly, attempt to flood the network with malicious requests to bring the network connection down. However, attacks against clients attempt to disrupt the service to a specific host.

Based on the technique of launching DoS attacks, Hussain et al. [206] have classified this techniques into two schemes: (a) The first scheme exploits vulnerabilities on the victim by sending crafted malicious packets to crash the victim; (b) the second scheme floods a vast volume of bogus packets to overwhelm the victim resources. CERT recommended that first scheme can be easily prevented by patching the vulnerabilities, while the second one is more difficult to prevent because the inter-networking systems are open community and the server (the victim) is center to provide services.

However, DoS and DDoS attacks can be classified broadly into two categories: (i) flooding attacks, and (ii) non-flooding (vulnerability) attacks.

4.2.1 Flooding DoS Attacks

These attacks work by sending a vast number of messages whose processing consumes some key resource at the target. Complex messages, such as messages require generating or verifying digital signatures or computing exponential operations consume much of CPU time, large size messages take up bandwidth, and initial communication messages with new clients take up memory. The strength of flooding attacks lies in the volume rather than the content of the attack. This has two major implications:

1. Attackers can send a variety of packets. The attack traffic can be made arbitrarily similar to legitimate traffic, which greatly complicates defense.

2. The volume of traffic must be large enough to consume the victim's resources. The attacker usually has to control more than one computer to generate the attack traffic. Bandwidth attacks are therefore commonly DDoS attacks.

Peng et al. [290] defined the impact of DoS attack by the term *attack power* which represents the level of resources consumed at the victim by the attack. Two parameters namely: *the traffic volume* and *the level of resources consumed per packet*, are consisted of the attack power. The first parameter refers to the number of packets sent in a certain period, while the second parameter is the level of resources (memory or CPU time) consumed per packet.

Following some examples of popular flooding DoS attacks:

TCP SYN Flood

The SYN flood attack [102] takes an advantage of the flaw of the TCP three-way handshake, that the server needs to allocate some memory for every incoming SYN packet regardless of its authenticity. The attacker sends SYN packets with unreachable source IP addresses. The victim server opens a half connection and stores the request information into the memory stack and waits for the configuration from the host that sends the request. As the request is not confirmed, the server will keep a state information of that request in the stack. Since the source IP address is invalid, the server will never receive a configuration. This will cause the server open many half connections until it becomes overloaded and is unable to serve any incoming requests. The impact of this attack is to exhaust the server memory.

Several techniques were proposed in order to countermeasure the SYN flood attack such as SYN Cookies [344] and CISCO techniques using Cisco IOS software [121].

ICMP Flood

The ICMP protocol is used to diagnose the network status. The *smurf* attack is a type of ICMP flood, where the attacker sends forged ICMP echo request to the network broadcast address of the vulnerable network and the request is forwarded to all the host within this network. Then, all of those hosts will send an ICMP echo request to the victim. This rapidly at least congests or disrupts the victim machine. In [10], several solutions were discussed.

UDP Flood

An attacker can carry out the UDP flood attack by sending a large number of UDP packets targeting random number of ports on the victim system. As a result, the victim system will check for the application listening at that port. When it realizes that there is no application is listening at that port, it will reply with an ICMP destination unreachable packet. Thus, if enough UDP packets are sent, the victim will be forced into sending many ICMP packets, leading it to be unreachable by the other hosts. The solution of this attack is deploying firewalls at critical points in the network to filter out unwanted traffic.

SIP Flood

Session Initiation Protocol (SIP) [308] is a widely supported open standard for call setup in VoIP. Ignoring the details of SIP signaling process, when A wants to call B, he will send an *INVITE* packet to B. Generally, this packet is sent to A's SIP proxy server, which will look up the address of B's SIP proxy server and send an *INVITE* packet to that proxy. When B's SIP proxy receives the *INVITE* packet, it will pass it to B's registered address and B's phone will ring. After this, either B picks up the phone to start the conversation or there is no answer.

In [322, 321, 108] an attack scenario was mentioned, where an attacker can flood the SIP proxy with many *INVITE* packets with a spoofed source IP addresses. Also, the attacker can use non-spoofed source IP addresses by launching his attack from a bot-net in order to avoid anti-spoofing mechanisms. This attack can effect both the SIP proxy servers and the callees. The resource of SIP proxy servers will be depleted by processing the *INVITE* packets. Furthermore, the network will be congested also. However, the SIP proxy server will not be able to provide VoIP service. This attack will also overwhelm the callee by the forged calls making them unreachable.

Additionally, several possible DoS attacks were mentioned in [6, page 25].

HTTP Flood

In generally, an HTTP flood refers to an attack that bombards Web servers with HTTP requests. One possible successful DDoS attack can be done by instructing a bot-net to send an HTTP request to the target Web site, then parse the replies and follow the links recursively. This attack is called recursive HTTP-floods. The similarity between the attacker traffic and the normal traffic makes it extremely difficult to filter such kind of attack.

4.2.2 Non-flooding DoS Attacks

Attackers launch vulnerability attacks by sending a few specifically crafted messages that exploit a vulnerability in the target application. This vulnerability is usually an implementation bug of a software or a bug in a default configuration of a service[1]. These attacks can be defended by patching the known vulnerability.

Ping of Death [7]

This is carried out by exploiting TCP/IP flaws. Normally, TCP/IP allows a maximum packet size up to 65536 octets. If an attacker send an ICMP Echo request packet that is much larger than the maximum IP packet size, the victim server, who receives these packets, cannot reassemble the packets and may be terminated.

Teardrop [103]

Teardrop is an attack tool that exploits a vulnerability in some implementations of TCP/IP fragmentation reassembly code. The vulnerable machine does not properly handle overlapping IP fragments. The impact of this attack is, an attacker can

[1] For example, some implementation of the 802.11 wireless access protocol have a vulnerability that allows an attacker to deny service selectively to one user in the wireless network or promiscuously to all of them. In effect, the attacker can send a packet to the wireless access point that claims to be from another user and that indicates that the user is finished and essentially wants to "hang up" [66]. The wireless access point then no longer recognizes communications from the targeted user. That user can reestablish communications with the access point, but the attacker can shut it down again in the same way.

remotely crash the vulnerable machine by sending two fragments that cannot be reassembled properly by manipulating the offset value of packets.

Echo/Chargen

Chargen (character generator) is a service designed to generate a stream of characters and is used primarily for testing purposes. An attacker spoofs a number of connections to various hosts' Chargen ports. If both services (Echo and Chargen) are enabled, an attacker can form a loop by pointing the spoofed echo sessions at the chargen service. This will cause a huge amount of data to be passed in an endless loop that will burn the network bandwidth and CPU cycles.

DNS Reflector Attack

An attacker sends a large number of requests to a name-server using a spoofed source address. The name-server will respond to this spoofed address (the victim). The responses of the name-server can be significantly larger than requests, which is potential for bandwidth amplification. The impact of this attack is bandwidth consumption of the victim and intermediary name server's network [220, 343].

4.2.3 DoS/DDoS Attack Tools

There are several tools that can launch DoS/DDoS attacks. In Table 4.2, we give a conclusion of different tools and the attacks, that can employ.

4.3 DoS Defense Mechanisms

Broadly, denial of service defense mechanisms can be categorized into four strategies [268, 25]: *detection*, *prevention*, *response*, and *tolerance*. Following, a brief review of these strategies.

4.3.1 Protection Techniques

Protection mechanisms attempt to prevent adversaries to perform DoS attacks. In other words, DoS protection mechanisms aim to stop attacks before they actually cause damage. Protection mechanisms include, but not limited to, Packet filtering and proof of work.

Packet Filtering

In order to hide their original IP addresses, DoS attacker rely on spoofed IP addresses. Filtering mechanisms aim to prevent DoS attacks with spoofed source addresses, by verifying the source addresses before providing the service and dropping the packets with false source addresses. For example, the CAPTCHA [345] is a

Tool name	Employed attacks
Trinoo (or Trin00) [154]	distributed SYN DoS
The Tribe Flood Network (TFN) [156]	DDoS attack tool, possible attacks: ICMP flood, SYN flood, UDP flood, and Smurf attacks.
Stacheldraht [155]	DDoS tool, possible attacks: ICMP flood, SYN flood, UDP flood, and Smurf attacks.
Trinity	Flooding attacks like; UDP, fragment, SYN, RST, ACK, etc.
Shaft [152]	packet flooding attack.
Tribe Flood Network 2K (TFN2K) [52]	Complex variant of TFN, possible attacks include flooding as in TFN, and malformed attacks such as in the Teardrop and Land attacks.
MStream [157, 105]	DDoS tool, it uses spoofed TCP packets.
Agobot and Phatbot	possible attacks SYN flood, UDP flood and ICMP flood.
Knight [104]	IRC-based DDoS attack tool, provides SYN attacks, UDP flood and an urgent pointer flooder [8].

Table 4.2: DoS/DDoS Tools

technique proposed to combat IP spoofing attacks. However, this technique is appropriate for communications between human user and computer, but it is not appropriate for communications between computer to computer. Several packet filtering approaches have been developed such as; *ingress/egress filtering* [170], *Route-based filtering* [283], *source address validity enforcement (SAVE) protocol* [239], *passport* [242], etc.

Proof of Work

This mechanism is proposed to counterbalance resource usage between a client and a server. One important example is the client puzzles technique [216]. Client puzzles counterbalance computational usage between client and server by forcing the client to solve a computational puzzle before attending to request. An attacker attempts to flood the server with a large number of bogus request will suffer from solving a huge number of puzzles, causing him to spend a lot of his computational resources.

Similar to the packet filtering technique, the proof of work approach helps for defeating IP spoofing attacks. Even though the attacker attempts to send a large number of requests or to flood a large number of bogus solutions to cause the server to waste resources by generating a large number of puzzles or by verifying bogus solutions, this attack is less effective because generation of puzzles and verification of solution are cheap operations. However, proof of work technique is good to defend

against computational (memory and CPU) attacks, but the major weakness of this approach is in defending of bandwidth depletion attacks.

4.3.2 Detection Techniques

It might be not enough to implement only prevention mechanisms to defend DoS attacks, due to the complex nature of DoS attacks. As an important procedure to direct any further actions, is deploying some detection mechanisms. The goal of detection mechanisms is to detect attacks when they occur. Response mechanisms depend on the attack information discovered by detection mechanisms for countering the attack. Some mitigation mechanisms depend on the fact that the attack is ongoing, in order to initiate the mitigation process. Since the attack traffic, in most cases, looks very similar to legitimate traffic, there is a high risk that the detection mechanism mistakes legitimate traffic as attack traffic. This is called a *false positive* which is a very serious concern of DoS attack detection mechanism.

Generally, there are two broad classes of detection techniques for identifying malicious actions. the first class is called *signature-based detection*, which is based on some features of attacks. The second class is called *anomaly-based detection*, which models the behavior of normal traffic, and then reports any anomalies.

Signature-based Detection

These mechanisms study the known DoS attacks to identify their unique characteristics that differentiate these attacks from normal user activities, and build a database of known attack characteristics (which are called attack signatures). The signature-based detection mechanism monitors the activity in the network for the presence of these signatures, if there is a match the suspicious activity will be removed. In order to maintain a low rate of false-positive alerts, the signatures need to very precise. The major drawback of this approach is that only the known attacks can be detected, while new attacks or some variations of old attack will not be detected.

Several signature-based detection approaches have been proposed such as: (1) Bro [288] network IDS, which is a real-time IDS passively monitors the network; (2) Snort [306], the open-source light-weight network IDS and prevention tool; (3) some approaches used spectral analysis of attack flows in order to identify DoS attacks [111, 207, 208]; and (4) in [224] a signature-based detection method was described using Kolmogorov complexity.

Anomaly-based Detection

Rather than profiling the signature of known attacks (as signature-based methods do), anomaly-based attack detection mechanisms analyze the normal behavior of legitimate users of the system and aim to detect attacks by identifying significant deviation from the normal behavior. The advantage of this approach over the signature-based detection approach is that anomaly-based mechanisms can discover

previously unseen attacks. However, the change of protocol specification and the variety of user applications make a challenge for the anomaly-based detection. The use of a tight threshold for legitimate behavior may wrongly detect normal behavior as malicious (false positive), while a loose threshold may lead to many attacks go undetected (false negative).

As an examples of anomaly-based detection mechanisms are: MULTOPS [184], D-WARD [269, 271] and SIM [289]

4.3.3 Response Mechanisms

The aim of the attack detection mechanisms mentioned above is to isolate the attack traffic. This should be done in a timely manner, in order to initiate the further actions to counteract the attack. After detecting the attack, usually response mechanisms are initiated to remove or reduce the attack impact. Response mechanisms include the following techniques:

Filtering and Rate-limiting

This technique uses the characteristics of the suspicious traffic provided by detection mechanism to filter or rate-limit attack traffic. This technique is considered most practical response since it requires less efforts for implementation. However, the challenge for designing an effective techniques is how to decide the suspicious traffic and to find a good balance between letting some attack traffic through and harming some legitimate traffic. An examples of these approaches such as: the aggregate-based congestion control (ACC) [253] for controlling high bandwidth aggregates in the network. Another approach is *StopIt* [243] which is a filter-based DoS defense framework aims to stop the undesired traffic intended to a receiver without inflicting damage on legitimate hosts sending traffic to that receiver.

Attack Source Traceback

When an attack has been detected, an ideal action would be to block the attack traffic at its source. Tracing IP packets to its source is not an easy task due to two reasons: (1) IP addresses can be spoofed, and (2) stateless nature of IP routing, where routers usually know only the next hop. The goal of this scheme is to traceback the suspicious traffic to the source of attack and then apply the law enforcement. In order to support the attack source identification, several approaches were proposed. Savage et al. [315] itroduced a probabilistic packet marking (PPM) scheme for tracing back the IP source. Another approach is SPIE (Source Path Isolation Engine) which is proposed by Snoeren et al. [325] to trace individual packets.

Capability

This scheme emerges from the fundamental problem of the Internet with regard to DoS attacks, which is the server has no control over who can send how much to it. There are several flow control and congestion control mechanisms that have been already implemented. However, the malicious clients can simply ignore the congestion and flow control signals and send traffic at the maximum possible rate. Capability schemes aim to provide mechanisms enable the server to stop such malicious hosts. An example is a Stateless Internet Flow Filter (SIFF) [352] which aims to selectively block undesired flows to reach the server. Traffic Validation Architecture (TVA) [353] is another example, which aims to limit the impact of flooding DoS attacks.

4.3.4 Tolerance Mechanisms

Tolerance mechanisms do not rely on the detection techniques and focus on minimizing the impact of DoS attacks to provide a better quality of service during the attack. These mechanisms can be categorized into several categories: congestion and policing, fault tolerance and client puzzles.

Congestion and Policing

The congestion mechanisms able to eliminate the effect of bandwidth DoS attacks. Several proposals were proposed in order to reduce the impact of bandwidth flooding attacks such as the *Re-feedback* techniques [87] and the *NetFence* [244].

Fault Tolerance

Fault tolerance mechanisms aim to achieve high availability of the system. As a DoS mitigation mechanism, fault tolerance mechanisms can be implemented by replication of service or multiplication of the resource used by the service. Fault tolerance mechanisms are very effective against DoS attacks, however they are costly to implement and there may also be a wasted resources during the non-attack period. As an example for fault tolerance schemes is [262] which proposed a capacity overprovisioning mechanism to maintain sufficient QoS of the network link during overloading attacks.

Client Puzzles

Client puzzles are usually cryptographic puzzles aim to provide resilience to the servers by counterbalancing resource using between clients and servers. The client is asked to solve a puzzle generated by the server before gaining access to a service. The concept of client puzzles are first introduced by Dwork and Naor [164]. Then several proposals [216, 163, 51, 178] have been followed to address the requirements of defense against DoS attacks. In the next section, a brief description of client puzzle mechanisms will be presented.

As this part is limited to the analysis of DoS attack defenses in key establishment protocols. In particular, we focus on flooding attacks that consume the server's computational resources (CPU and memory). Additionally, we limit ourself to those cryptographic protocols that implement cryptographic puzzles as a DoS-resistance mechanism to counterbalance the resource usage between clients and the server, as well as to penalize attackers attempting to flood the server by bogus requests. As mentioned above, there are several DOS defense mechanisms based on various cryptographic techniques that aim to protect and assist the server from DoS attacks such as resource exhaustion attacks. The following section 4.4 briefly introduces cryptographic based DoS defense mechanisms.

4.4 Client Puzzles based DoS Defenses

The concept of using cryptographic puzzles (or client puzzles) was firstly introduced by Dwork and Naor [164] to combat junk mails and control access to shared resources. Later this approach was extended by Juels and Brainard [216] to thwart DoS attacks in network protocols. The concept of client puzzle as a DoS resistance mechanism has been implemented in key establishment protocols like the Host Identity Protocol (HIP) [274]. A study of DoS-resistance in key establishment protocols by Smith et al. [324], which reviewed three key establishment protocols that implement DoS-resistance mechanism.

Many client puzzles have been proposed based on different underlying techniques. However, they must fulfill the good puzzle properties described in [216, 51]:

1. The puzzle generation as well as the solution verification is inexpensive for the server.
2. The Puzzle's level of difficulty can easily be adjusted from trivial to impossible.
3. The puzzle is solvable on most common client hardware.
4. It should be not possible to precompute solutions to the puzzles.
5. By issuing a puzzle, the server does not need to store any state.
6. Knowing the solution of one or more clients is of no benefit for another client to solve the puzzle, thus the same puzzle may be issued to several clients.
7. A client can reuse the puzzle by creating instances of it.
8. Another important property of a good puzzles is *non-parallelization*, which states whether the solution can be computed in parallel or not [342].
9. Puzzle fairness and minimum interference [27]. *Puzzle fairness* represents that computing the puzzle solution will take the same amount of time for all clients, regardless of their computational power. While *minimum interference* means that solving the puzzle should not interfere with the client's normal tasks.

After introducing the characteristics of constructing good puzzles, we provide a brief survey of some existing proposals of client puzzle.

Hash-based Reversal Puzzles

Hash-based reversal puzzles were introduced by Juels and Brainard [216] to protect the server against SYN flooding attacks. The server constructs the puzzle using hash function and sends it to the client. To solve this puzzle, the client needs to calculate a reverse one-way hash value of the puzzle. The client performs a brute-force search on the puzzle bits by hashing each pattern until it finds the solution. If the puzzle has k-bits, the client requires to perform in average 2^{k-1} hash operations and in worst-case would be 2^k hash operations. The difficulty level of the puzzle can be easily adjusted by increasing or decreasing the value sent to the client in the puzzle. To verify the solution sent by the client, the server requires to perform only one hash operation.

Another similar puzzle construction was proposed by Aura et al. [51], which consists of a time value (t), server nonce (N_R), and a difficulty parameter (k). The client must find a value x that when hashed with N_R produces a digest output whose first k-bits are zero. For verifying the solution, the server first checks whether t and N_R are recent or not. Then, the server verifies the solution x by hashing N_R to produce an output whose first k-bits are zero. This puzzle construction is of similar complexity as Juels and Brainard puzzle.

The fast and simplicity of generating and verifying the puzzle by the server are the major strength of this mechanism. However, the drawbacks of this mechanism are that it is parallelizable and its granularity is exponential as well as the time of finding a solution is probabilistic in nature and depends on luck (a client may find a solution after the first try or after 2^kth try).

Hint-Based Hash Puzzles

To improve the granularity of the hash-based reversal puzzles, Feng et al. [107] proposed another puzzle constructed called *hint-based hash puzzle* to allow the granularity to be linear. In this mechanism, the server provides extra information called *hints* attached to the puzzle. Instead of checking every possible solution, the client searches for a solution within a range of given hints.

Similar to the above approach of Juels and Brainard, the server spends one or two hash operations for generation and verification of the puzzle. Although the client computational cost could be up to k hash operations. The advantages of this approach are fast construction and verification as well as linear granularity of the puzzle difficultly level. However, the disadvantage is that it is still parallelizable.

Time-Lock Puzzles

This technique is developed by Rivest et al. [305] and based on the notion that the client has to spend some pre-determined time performing repeated squaring

to search for a solution. The server estimates the performance of the client by the number of squaring operations the client can perform in a give period and determine the amount of time it expects the client to spend solving the puzzle.

Since the squaring operations must be calculated sequentially, this technique is non-parallelizable. Furthermore, the granularity of this puzzle is linear because the period of solving the puzzle is easily controlled and determined by the server at the construction time of the puzzle. However, the high-computation of construction and verification of the puzzle is the major drawback of this approach.

Diffie-Hellman Based Puzzles

The DH-based puzzle scheme was proposed by Waters et al. [347]. The generation of the puzzle requires one modular exponentiation which is an expensive operation. The verification of solution, in general, requires also one modular exponentiation, but it can be minimized using table lookup. The solution of a certain time slot can be precomputed by the server and verified via table lookup. The client searches for a solution within a range given by the server as a hint by testing each possible candidate value in the range until finding a correct solution.

The expensive construction of the puzzle is the major drawback of this scheme, while the linear granularity is an advantage. However, this scheme does not support the non-parallelizability property.

Trapdoor RSA-based and DLP-based Puzzles

To overcome the weakness of puzzle construction of hash-based scheme, Goa [178, 179] proposed two puzzle schemes based on trapdoor functions named *trapdoor RSA-based puzzles* and *trapdoor DLP-based puzzles*. To reduce the computational cost of the puzzle construction the server pre-computes some parameters before starting the protocol. The initial pre-computing requires the server to perform one modular exponentiation plus one modular multiplication for construction the puzzle in case of trapdoor RSA-based puzzles and it is reduced to only one modular exponentiation in case of trapdoor DLP-based puzzles. The server also pre-calculates the solutions and stores them. This initial pre-computing is a disadvantage of these schemes. The client requires to find the solution within a given range as a hint. The granularity of this approach is linear. The puzzle can be distributed and computed in parallel, thus the scheme does not support the non-parallelizability.

Cryptographic Chaining Puzzles

The common problem of the aforementioned approaches is that none of them provide the characteristic of non-parallelizability except time-lock puzzles which suffer from an expensive construction operation for the server. A promising technique for preventing puzzles to be computed in parallel is *chaining* technique, since it requires the previous value in order to construct the next consecutive one. There are two

constructions using the chaining technique based on cryptographic hash functions introduced by Ma [251] and by Groza [192].

Ma's Chain Puzzles

Ma [251] proposed a scheme called password puzzles (PP) to be used in IP layer, which are a hash-chain-reversal puzzles. The construction of the puzzle begins by choosing a random number h_0 as an initial value. Then, the server applies a one-way function to h_0 repeatedly to generate the desired hash chain $h_0, h_1, ..., h_k$ where $h_{i+1} = hash(h_i)$ and k is the length of the chain. According to the author, this computation has an advantage for the server, that it can store the entire chain for further use. As the server knows the corresponding solution in advance, this reduces the cost of verification of the puzzle solution to a single table lookup.

In order to solve the puzzle, given a puzzle challenge consists of the last value of the hash chain h_k and an index value k, the client needs to obtain the entire hash chain by computing a hash reversal starting form index k back to the start point h_0.

This scheme is a non-parallelizable technique owing to the hash chain mechanism. The cost of verification is k hash operations similar to the construction. However, storing every value of the entire hash chain by the server despite the benefit of minimizing the cost of verification, it makes the server susceptible to memory exhaustion attacks.

Groza & Petrica's Chain Puzzle

In this scheme [192], the puzzle is constructing from a hash chain of random numbers. The server chooses two state-dependent random numbers ρ and r, then concatenating them to obtain a value σ. By double hashing σ_0, the first output P_0 will be obtained. The rest of the puzzle is generated by XORing two new state-dependent values with hashed output of σ from the previous state. Then, the puzzle challenge sent to the client would be a series of pairs $[(P_0, r_0), (P_1, r_1), ..., (P_k, r_k)]$, where $k \geq 1$ is the length of the hash chain. For solving the puzzle, the client needs to perform a forward process of reconstructing the hash chain by searching for ρ_i values, which is $\sigma_i = \rho_i \parallel r_i$.

In order to construct or verify the puzzle, the server requires to perform three hash operations per state for generating the hash chain. Thus, the server is susceptible to resource exhaustion attacks. In addition, the amount of computation required by the client to solve the puzzle is similar to the amount of computation of constructing the puzzle by the server. This violates the property of designing a good puzzle that states; "client puzzles should be easily constructed and verified by the server and hard to be solved by the client".

Threshold Puzzles

Threshold puzzles were proposed by Bocan [75, 76]. The design goal of threshold puzzle is that, the puzzle difficulty level is usually based on the server commitment. An attacker with high computational power is able to solve puzzles much faster than legitimate clients. Since the difficultly level, in several puzzle construction techniques, increases exponentially, puzzles with high level difficultly may not be solvable by legitimate clients. This leads to a DoS attack against legitimate clients.

Threshold puzzles propose a mechanism that the server can keep estimating the time that the client took to solve the puzzle. However, the generating process of the puzzle depends on hash operations and generating periodic nonces. In addition, the server requires to calculate the time that the client took to solve the puzzle and compare it with the given estimated time. In the other side, the client requires 2^{k-1} operations in average to solve a puzzle of difficultly level k.

This design does not fulfill the non-parallelizability property or reduce the granularity to be linear, but claimed to limit them by bounding the level of difficultly and provide the minimum response time. Furthermore, the server needs to do several calculations such estimating the minimum time and calculate the time that the client took to solve the puzzle and compare it with the estimated time.

Subset Sum Cryptographic Puzzles

This scheme is based on a known cryptographic problem called *subset sum* or *knapsack* problem. Knapsack problem is like: given a set of items, each of them with a weight and a value, the solver requires to pick a number of each item to include in knapsack so that the total weight is less or equal to a given limit and the total value is as large as possible. The subset sum puzzle is introduced by Tritilanun et al. [342], the main goal of this scheme is to provide a non-parallelizability characteristic along with simple construction and verification.

The construction of the puzzle requires that the server performs n hash operations as pre-calculation and one hash operation for the generation, while the verification requires approximately two hash operations. The scheme provides the non-parallelizability characteristic and it has polynomial granularity. Although the subset sum is NP-complete problem, there are several algorithms can solve it. The most efficient known algorithm that can break the subset sum problem is the LLL (or L^3) algorithm [236], which a lattice reduction algorithm. However, implementing LLL algorithm requires clients that have strong computational power.

Guided Tour Puzzles

The guided tour puzzle scheme is proposed by Abliz and Znati [27], which requires that the client should follow some rules in order to retrieve the answer of the puzzle. The authors claimed that the guided puzzle scheme achieves all of the aforementioned properties of a good puzzle among them non-parallelizability, puzzle fairness and minimum interference.

The puzzle is a set of special nodes called *tour guides*. The client is asked to complete a tour by visiting those nodes (tour guides) in a certain sequential order. For constructing the puzzle, the server needs to maintain some shared secret keys with each tour guide and these need to be stored. Also for reducing the verification of the puzzle answer, the server calculates the answers in advance and store them using bloom filter. Then, the server needs a single hash to generate the puzzle and a single lookup for verification of the answer of the puzzle.

The scheme supports the fundamental properties of designing good puzzles including: non-parallelizability, fairness, etc. However, the involvement of the server of generating the puzzle by manipulating and storing several secret keys and precomputing the answers, this can be seen as drawbacks of this scheme.

Game Theory

Several game-theory-base puzzles were proposed in [168] and [275]. In the later, two new properties were also suggested namely; hidden difficulty and partial solution. hidden difficulty means that the puzzle difficultly should not be determined without paying a minimal computation cost. While a partial solution represents that submission of partial solutions is possible without increasing the verification time.

The proposed puzzles were based on the previous mentioned puzzles of the hash-reversal puzzle and hint-based puzzle. In this puzzle proposal a special emphasize was given to the hidden difficulty property. Puzzle generation takes maximum four hash operations, while verification takes maximum six hash operations. This scheme does not support non-parallelizability requirement.

In summary, we have discussed several client puzzle proposals. In the following table (Table 4.3[2]) briefly summarizes the key strengths as well as the drawbacks of the above mentioned client puzzle proposals.

4.5 DoS Resistance Strategies and Techniques in Authentication Protocols

Authentication and key establishment protocols are the foundation of secure communication in volatile environments. The goal of authentication and key establishment protocol is to enable two parties to authenticate each other and establish cryptographic keys using insecure environments. These protocols are usually vulnerable to DoS attacks that exhaust the system resources as they are required to perform expensive operations.

Several authentication and key establishment protocols with DoS-resistance have been proposed. A detailed review study of Dos-resistance strategies and techniques in key establishment protocols was presented in [324, 340]. In this section, we briefly

[2] This table is based on and extends [340, Table 2.1] and [342, Table 1].

Client Puzzle	Strengths	Drawbacks
Hash-based reversal	Simple and fast construction and verification	Exponential granularity, Parallelizable
Hint-based Hash Reversal	Simple and fast puzzle construction and verification, Linear granularity	Parallelizable
Time-lock	Non-parallelizable, Linear granularity	High computational cost for construction and verification
DH-based	Linear granularity, Cheap verification	Expensive construction, Parallelizable
Trapdoor RSA-based and Trapdoor DLP-based	Cheap verification, Linear granularity	Parallelizable, Expensive construction
Hash chain	Cheap verification, Linear granularity	Parallelizable, Expensive construction
Threshold puzzle	Cheap verification, Limiting difficultly and response time	Extra Calculations requirement, Parallelizable, Exponential granularity
Subset sum puzzle	Non-parallelizable, Cheap verification	Polynomial granularity, Precomputation requirement
Guided tour	Non-parallelizable, Puzzle fairness, Cheap construction and verification	Pre-computation requirements, Maintenance of several keys
Game theory	Cheap construction and verification, Hidden difficulty	Parallelizable

Table 4.3: Client Puzzle Summary

present those strategies and techniques, then we list the key establishment protocols with DoS-resistance a long with DoS-resistance that it implements.

4.5.1 DoS Resistance Strategies

Usually DoS attacks utilize some vulnerabilities of the underlying protocol. An example flooding attacks, which aim to exhaust the resources of the target victim by sending a large number of bogus requests. In case of authentication and key establishment protocols, an attacker may flood the target server with a large number of requests forcing it to compute expensive cryptographic operations in order to consume its CPU cycles.

However, protocol engineering community has realized the requirement that authentication and key establishment protocols should achieve a level of DoS-resistance. Hence, number of design strategies have emerged. For example, Leiwo et

al. [234] and Meadows [260, 261] proposed several design principles for developing DoS-resistant protocols. Those strategies can be broadly classified into three classes [324]: preventing CPU exhaustion, preventing memory exhaustion and gradual authentication.

Preventing CPU Exhaustion

Normally, the server responds to many clients requests. In order to prevent the computational resources of the server and prevent clients from launching flooding attacks, the client workload should consume those resources used to carry out the attack [234]. Achieving this goal requires either increasing the cost of computation to the client or reducing the cost of computation to the server. This can be carried out by artificially increasing the computational cost of the client (e.g., solving a puzzle) or by having the client perform computations on behalf of the server. Client puzzles [305, 216, 51, 107] and a client-aided RSA computation [99] are example techniques of this strategy.

Preventing Memory Exhaustion

To counteract the memory attacks, a direct strategy is to reduce the memory usage for the server during the protocol run. This can be insured by allowing the server to maintain a stateless connection while the authentication process has being not completed. Using *Cookie* strategy is an example defending technique preventing memory exhaustion attacks [235]. Nowadays, cookies is widely used by authentication protocols designers. There are many authentication protocols which use cookies as a defense against memory exhaustion attacks, such as, Photuris [217] and Internet Key Exchange (IKE) [199, 256].

Gradual Authentication

A client that requests a service must be authenticated at some point during the protocol run, strong authentication of the client might be used to carry out a DoS attack. This happens due to the use of expensive operations to provide strong authentication. A proposed strategy to resolve this problem is to use weak authentication at the starting of the protocol and gradually increase the authentication level ending up with a strong authentication at the completion of the protocol [260]. The Just Fast Keying (JFK) [32] protocol is an example protocol that uses gradual authentication. JFK starts with using a nonce and an authenticator as a cookie, then using MAC and finally completes the protocol with a signature verification.

4.5.2 DoS Resistance Techniques

The primitive DoS-resistance techniques used to implement the aforementioned strategies are; cookies, client puzzles, client-aided computation, and gradual authentication. A brief description of these techniques is given as following.

Cookies

While receiving a service request by the server, it issues unpredictable time variant data that allow it to remain stateless and gradually initiate the authentication process to the client. This data are called cookies. Cookies have been used widely as a DoS-resistance technique, as an example, in defending SYN flooding DoS attacks [235] and in Photuris protocol [217].

Usually, a cookie consists of some connection parameters cryptographically transformed using time variant secret such as: keyed hash of the client IP and nonce. For preventing the memory exhaustion attacks and remain stateless, the server may include any state required parameters in the cookie which can be returned by the client in the next message. The state required data can be verified by the server when the received cookie is valid.

Puzzles

Puzzles are typically constructed using cryptographic functions (one-way hash functions). They allow the client to prove to the server that an amount of resources has been expended, in order to ensure his/her willingness to commit resources. At first, puzzle was introduce to control junk email [164], later have been extended to be used to protect authentication protocol from DoS attacks [216, 51]. Several schemes have been proposed for constructing puzzles, which already explained in section§4.3.

Client-Aided Computation

The idea of client-aided computation is to have the client to perform computations on behalf or to ease the computation burden of the server. Usually, such computations are performed before the client fully authenticated, thus the client computations at this time should be restricted to those that can be verified correctly. This technique is implemented in the client-aided RSA protocol [99] which is a modification of SSL protocol.

Gradual Authentication

Strongly authenticate clients at the beginning of the protocol using computationally expensive operations might be costly to verify these operations also for the server. This can cause the server goes under DoS attack. In order to remedy this situation, the gradual authentication is proposed [260, 261]. Gradual authentication provides a mechanism for weakly authenticating the client before performing expensive operations. The idea is to use a weak authenticators likes cookies or client puzzles at the beginning of the protocol and moving to strong authentication gradually at the protocol completion.

Cookies, client puzzles, message authentication codes and release of hash digest preimage can be considered as forms of weak authenticators. The key strength of these techniques of weak authentication is that their verification is cheap for the

server comparing to their fabrication by the attacker. Furthermore, the verification of the gradual authenticators such as client puzzles remains cheaper than the verification of the signature schemes that can minimize the verification cost to the server.

As an example of authentication protocols that employed the gradual authentication; the Just Fast keying and the modified Internet Key Exchange (IKE) protocols.

Many authentication and key establishment protocols have been proposed to provide DoS-resistance goal by implementing some of the above mentioned DoS resistance strategies and techniques. In [324, 340], seven authentication and key establishment protocols that implement DoS-resistance mechanism were discussed. They are: Photuris [217], Modified IKE [256], IKEv2 [218], JFK [32], HIP [274], Lee & Fung [233] and Client-Aided RSA (CA-RSA) [99].

In the next section, we will give a brief overview of the formal methods that have been proposed for analyzing the DoS-resistance mechanisms.

4.6 Analysis of DoS Attacks

In chapter 2 we have presented a detailed survey of formal approaches for analyzing security protocols. Most of the mentioned approaches are suitable only for analyzing secrecy and authenticity requirements. Which are typically expressed as safety properties and verified in presence of of the *Dolev-Yao*-like attacker model, who has a complete control of the network. However, security requirements such as *availability*, considerably were less studied.

Availability is quantification of the alternation between proper and improper service, and is often expressed as the fraction of time that a system can be used for its intended purpose during a specified interval of time or in a steady-state [278].

Unlike security requirements such as secrecy and authenticity, which are expressed as safety properties, DoS threats need to be expressed as availability properties. AlTurki et al. in [36] addressed the challenges of specifying and verifying availability properties. Since DoS resistance properties require to be expressed in degrees, such that; it is reasonable to ask whether a given DoS protection mechanism is more effective than other one. Consequently, a movement from boolean valued (true or false) requirements such as secrecy to *quantitative* requirements such as effectiveness in protecting against DoS attack is required. Another important issue is the Dolev-Yao intruder model [162], which is usually used as an attacker model for analysis of securecy and authenticity properties, is not entirely suitable for verification of DoS attacks. The strong assumption of Dolev-Yao model considers that the attacker has a complete control of the network. It means by definition, the intruder can always succeed in performing DoS attack. Furthermore, the Dolev-Yao model is too rough since the amount of resources available to the attacker needs to be estimated quantitatively.

4.6.1 Formal Approaches for Analyzing DoS-Threats

It has been mentioned that relatively few formal techniques have been proposed for analyzing and verifying availability requirements (such as a protocol being resistant to DoS attack) comparing to the works in formal methods for verification of secrecy and authenticity requirements. There are several works in literature that formally analyze DoS attacks. In this section, we give a brief description of those formal models that aim to analyze DoS threats. We categorize those works, based on their main formalisms, into the following groups:

General framework approach

The works of Meadows [260, 261] are the foundation of this approach and they are the first works that show the importance of availability analysis for cryptographic protocols. Meadows in [261] developed a *cost-based framework*, in which DoS attacks were examined in the context of the resource intensive task of authentication. This framework weights the cost to the defender against the cost to the attacker. The author described the possible integration scenarios of the proposed framework into the automated security analysis tools that were available at that time. A limitation of this framework is that the computational costs are not defined precisely, but consist instead of of a small number of discrete values. As described by Meadows herself, this framework is a "crude and ad hoc cost framework". Meadows's work has later inspired other cost-based approaches of DoS attacks analysis, including:

Lafrance and Mullins in [230] proposed to detect DoS attacks using a process algebra called Security Process Algebra (SPPA) and cost-based framework. They introduced an information flow property, called *impassivity*, which detects when an attacker, using his low-cost actions, causes interference on high-cost actions of other principals.

Smith et al. in [323] exploited Meadows's framework to analyze the Just Fast Keying (JFK) protocol [32], in order to demonstrate its DoS resistance mechanism.

Tritilanunt in [339, 341] used a refined model of Meadows's framework [261] to simulate DoS-resistant protocols. A quantitative evaluation was done by simulation of the developed Timed Coloured Petri Net model without having exploited the formal analysis capabilities of CPN Tools.

Game-based approach

Game-based analysis is proposed in [254] by Mahimkar and Shmatikov. Tow classes of DoS attacks, bandwidth consumption and resource exhaustion, were described. The analyzed protocols were specified as alternating transition systems (ATS) and their security properties were stated in alternating-time temporal logic (ATL) and verified using MOCHA model checker. As a case study, the authors analyzed a modified version of a key-exchange protocol (JFKr) that uses client puzzles. The interaction between the attacker and the server is modeled as a two-player strategic

game. The protocol is verified for fairness towards the clients and the attacker with respect to their solving of the client-puzzles.

Stochastic modeling approach

Madan et al. [252] introduced an interesting stochastic model for quantifying the availability for software systems. The system is formulated and analyzed in terms of an appropriate semi-Markov process (SMP). Starting with the SMP model, it is then possible to derive the embedded DTMC that involves only the considered state transition probabilities. After having computed the steady-state DTMC probabilities, the assumed sojourn time distributions for the model's states are used to compute the SMP's steady-state probabilities. This makes it possible to calculate the system's availability and subsequently perform parametric sensitivity analysis in order to examine the sensitivity of the computed availability. This analysis was not carried out within an automated analysis tool, so it requires stochastic modeling and analysis skills. Also, the performed system-level analysis does not take into account the resource expenditure for the considered states and thus it is not possible to evaluate the message processing costs for DoS threats upon a security protocol model.

Probabilistic-based approach

The underlying formalism of this approach is Markov models (DTMC, MDP, or CTMC). In what follows, we present two probabilistic based models. he first one used the probabilistic rewriting theory (*PMaude* system) as specification language and the second one used the probabilistic model checker for specifying and analyzing DoS attacks. Using first approach (*PMaude* system) approach, Agha et al. [29] have specified and analyzed the TCP 3-way handshaking DoS resistance. Similarly, AlTurki et al. in [36] have used probabilistic rewrite theories to model and analyze DoS protection capability of the ASV protocol. Using the second approach, probabilistic model checking, Basagiannis et al. [54, 55] introduced a methods for quantifying anylsis of DoS resistance properity. The authors analyzed the Host Identity Protocol (HIP) in order to validate their approach. The drawbacks of this approach are: (1) the costs are specific only for HIP protocol; (2) the analysis was focusing on DoS attacks against the initiator rather than the server; and (3) does not strictly follow the cost-based framework of Meadows.

4.6.2 Evaluation

To compare the aforementioned techniques that have been described in Section 4.6.1, we evaluate them using different criteria as they target different types of DoS attacks and use different modeling and analysis methods. Following a short discussion of the above approaches with respect to the evaluation criteria.

Specification and Verification Language

Readability and expressiveness of the specification language have a great importance for developing system models. Also, well developed mathematical verification techniques are capable to exploit a large number of state space of the system and can deal with different types of analysis such as quantitative analysis which is very important in the network security spectrum.

Here briefly we describe the specification language or the notation and how the DoS attacks are assessed in each approach of the aforementioned approaches in Section 4.6.1.

Meadows's cost-based framework [261] is based on a modification of Gong and Syverson's fail-stop model of cryptographic protocols [188] in which a protocol stops executing on message verification failures, and ascribes costs to each step of a protocol allowing an ongoing comparison of resource expenditure at both the initiator and responder. Protocols were specified in the framework using the Alice-Bob notation style and were annotated to include the steps associated with generation, verification, and acceptance of messages. Events associated with the generation, transmission, verification, and acceptance of messages were identified and assigned costs, for example computational costs.

A set of cost functions were defined to define the costs of actions of legitimate protocol participants. Similarly from the attacker point of view, a set of cost functions were defined in order to measure the cost of the attacker actions, because the attacker actions were treated independently from the legitimate protocol participants. That means that the attacker is not constrained to the same events or subject to the same cost functions as a legitimate protocol participant.

To assess DoS resistance of a protocol, the costs of actions of legitimate protocol participants were compared to the costs incurred by an attacker as a result of participating in the protocol. A DoS attack is then characterized by having legitimate participants expend more effort than a given threshold, specified by a tolerance relation in the framework.

Security Protocol Process Algebra [230], in this approach a process algebra (called Security Protocol Process Algebra (SPPA)) was used to specify the protocol. The analysis of DoS attacks is carried out by using SPPA and a cost-based framework to introduce an information flow property called *impassivity* which detects any case when an enemy process may cause interference, using its low-cost actions, on high-cost actions of other principals.

Game-base analysis, in this approach protocols are specified as Alternating Transition Systems (ATS) with a game-based formalism (which is a game variant of Kripke structure), and the desired security properties are stated in Alternating-time Temporal Logic (ATL) and verified using MOCHA model checker. The interaction between the attacker and defender is modeled as a two-player strategic game. The

defender's strategy is characterized by the difficulty level of the defense mechanism (e.g., difficulty levevl of puzzle generated by the server) that is generated and presented to the client, whereas the DoS attacker's strategy is characterized by the amount of effort it can invest in solving the puzzles.

Probabilistic Rewriting Theory [29, 36], in this technique a protocol is specified using probabilistic rewrite rules in order to develop a probabilistic model. Probabilistic rewrite rules are rewrite rules whose right-hand side result is not uniquely determined by the matching substitution for their left-hand side, but depends instead on the probabilistic distribution. The quantitative properties are expressed in query language called *Quantitative Temporal Expressions* (QuaTEx) which is motivated by the popular Probabilistic Computation Tree Logic (PCTL) and EAGLE. Once a QuaTEx formula and a desired degree of statistical confidence are defined, a number of Monte Carlo simulations of the rewrite theory is performed in order to verify the QuaTEx formula. In both papers [29, 36], the Monte Carlo simulations are performed using the Maude rewrite engine, and the statistical model checking of the QuaTEx properties is performed by the VeStA tool [30, 319], which has a Maude interface.

Probabilistic Model Checking approach [55] used the specification language of PRISM model checker to specify the protocol and the attacker model. PRISM language is a simple state-based language based on the Reactive Modules formalism and it is used to specify all types of models that PRISM supports: DTMCs, MDPs and CTMCs. In this approach, a protocol is expressed as a probabilistic reachability property that is automatically verified with respect to the generated DTMC.

Implementation or Automated Tools

Automated tools facilitate modeling and analysis of systems by exploiting the space state automatically and reporting about the specified system. Some of the previously described approaches have been implemented or used already existing automated tools and some others are not implemented and has not been automated. Following is a summary of the aforementioned approaches and their related automated tools.

Meadows's cost-based framework [261] has not been implemented. Only a description was given of how this framework can be integrated with some existing automated tools available at that time. Later, Tritilanunt in [339] using Timed Coloured Petri Nets (TCPNs) developed a refined model of cost-based framework for analyzing HIP protocol and used the simulation capabilities of CPN Tools in order to automate the analysis of that model.

Similarly, the SPPA [230] has no implementation or automated tool presented to automate the analysis.

Different from cost-based and SPPA techniques, probabilistic rewrite theory [29, 36] technique is supported with mature automated tools

The technique in [29, 36] used PMaude [30], a probabilistic extension of Maude rewrite system, for specifying the model and performing the Monte Carlo simulations. And for the statistical analysis of the quantitative properties, which was expressed in QuaTEx language, the statistical model checking called VeStA [319] which is supported by Maude interface, is used. Both tools are mature and are continuously improving.

Suitability of The Proposed Approaches

Denial of service (DoS) attack aims to deny access of legitimate users to shared services or resources. Based on the effect caused by the DoS attack, there are two basic types of DoS attacks: Bandwidth consumption and resource exhaustion (Memory and CPU). The following table (Table 4.4) shows the suitability of the proposed techniques, in Section 4.6.1, for analyzing each type of DoS attacks.

Approach	DoS attack type	Remarks
Meadows's Cost-based Framework	Resource exhaustion	The studied protocols are: STS [261], JFK [323] and HIP [339].
Security Protocol Process Algebra	Resource exhaustion	The studied attack is TCP SYN [230].
Game-based Approach	Both bandwidth consumption and resource exhaustion	The studied protocol was JFK [254].
Probabilistic Rewrite Theory	Both bandwidth consumption and resource exhaustion	TCP SYN was studied in [29] and ASV protocol was studied in [36].
Stochastic Modeling	Both bandwidth consumption and resource exhaustion	The analytical model presented in [252] is for quantifying availability requirements.

Table 4.4: Illustrates the suitability of the above approaches

Scalability

Most of the above approaches demonstrated an example of resource exhaustion DoS attacks except the game-based approach that has shown an example of each type. Furthermore, none of the approaches, as originally proposed, is suitable for analyzing DDoS attacks. This is due to the complexity of modeling coordinated attackers in particular in the case of using model checkers which have a general limitation, that is the state explosion problem.

However, Smith et al. in [323] demonstrated the impact of a coordinated group of attackers sharing the same value. If there are n attackers, he showed that, by

using Meadows's framework, the cost can be reduced by a factor of $\frac{1}{n}$.

Thus, analysis of distributed denial of service attacks (DDoS) is not an explored area and would be an open research issue.

4.7 Discussion and Conclusion

This chapter introduces the Denial-of-Service (DoS) and Distributed Denial-of-Service attacks by reviewing the fundamental related knowledge and showing some statistical information that emphasize the impact of DoS threats on todays Internet. DoS Defense mechanisms in general , and DoS resistance strategies and techniques in authentication protocol in particular also are described. We describe several client puzzle schemes focusing on their strengths and weaknesses. In addition, some formal methods for modeling and analysis DoS attacks are introduced.

It has been shown that DoS attacks have a great impact on the Internet traffic as well as the growing of bot-nets (attacker's army) is also significantly increased. In 2010 a large bot-net called *Rustock* has been observed, with more than one million bots under its control.

We mention that, DoS attacks can be categorized into two broad categories: (1) flooding attacks which are launched by sending a large number of malicious requests; and (2) vulnerability attacks which are launched by sending crafted requests to exploit a vulnerability in the victim. Several DoS attacks like TCP SYN, Ping of Death and so on, as well as DoS attack tools like Trinoo and TFN along with the attacks that can employ, have been introduced.

In term of DoS defense, there are broadly four mechanisms namely: (1) protection mechanisms; (2) detection mechanisms; (3) response mechanisms; and (4) tolerance mechanisms. Protection mechanisms that attempt to prevent DoS attacks before they cause damage. Those techniques include packet filtering and proof of work. Detection mechanisms which aim to detect attacks when they occur and provide an input information for response mechanisms. Detection mechanisms are classified into two main classes namely signature-based and anomaly-based detection. Response mechanisms aim to isolate the detected attack traffic. Those mechanisms include filtering and rate-limiting, attack source traceback and capability schemes. Finally, tolerance mechanisms which focus on minimizing the impact of DoS attack in order to maintain acceptable QoS during the attack period. Tolerance mechanisms include congestion and policing, client puzzle techniques.

Additionally, this chapter presents the client puzzles which mainly used as a defensive technique in several authentication protocols to counterbalance the resource usage between the server and the client. We describe puzzle construction and verification as well strength and weaknesses of each puzzle scheme are also identified. We also provide the strategies and techniques that can improve the resistance of authentication protocols against DoS attacks. Those strategies include preventing resource (CPU and memory) exhaustion and gradual authentication.

Finally, we present a through analysis of the state-of-art of formal methods for modeling and analysis of DoS-resistance attacks. Those methods are briefly described and evaluated based on some characteristics of the formalism including: specification language, implementation, suitability of which type of DoS attack and scalability of the method. The studied approaches have been categorized based on their mathematical main formalism, into these categories: general framework approach, game theory based approach, stochastic modeling approach and probabilistic-based approach.

To conclude, this chapter presents the fundamental knowledge of Denial of Service attacks including: DoS attack types, DoS attack tools, defensive mechanisms in authentication protocols against DoS attacks, DoS-resistance strategies and techniques in authentication protocols and a through analysis of formal methods that already proposed for modeling and analyzing DoS attacks.

Chapter 5

Analyzing DoS Attacks Through Simulation

Contents

5.1 Overview

It has been mentioned in earlier chapters that the increase in frequency, severity and sophistication of DoS attacks, makes them the major current threat of Internet nowadays. Distributed DoS attacks can generate a huge volume of unwanted traffic which can thwart the network connectivity. Additionally, due to the complexity of internetworking environments, it is hard to detect and mitigate such kind of DoS attacks. Generally, the impact of DoS attacks can be either rendering the service to become unavailable or causing a degradation in the QoS of the system. However, the effect of DoS attack depends on the protocols and applications in use. Thus, it is desirable to measure the resilience of systems against DoS attacks.

In order to protect against DoS attacks, several methods have been proposed. Boteanu et al. [81] had classified those methods into three classes. The first class is traffic filtering mechanism which uses traffic profiling or anomaly detection mechanisms (a detailed survey can be found in [270, 268]). The second class is based on influencing the resource trade-off in favor of the defender by modifying the relevant protocols such as TCP/IP protocols (proof of work mechanisms such as [216, 360] fall in this class of defense). The third class is based on queue management schemes. Queue management schemes can mitigate the residual traffic that could reach the end-server by evading the first two classes. Previously in chapter 4 §4.3, we have mentioned that queue management fall in tolerance mechanism defense which aims to minimize the impact of DoS attack in order to provide a better QoS during the attack duration.

In this chapter, we used several performance metrics in order to quantify DoS attacks impact of the service availability experienced during the attack on the network as well as to evaluate their active defense mechanisms. These measures are based on QoS and performance analysis metrics. In addition, the measurements are calculated for different applications (e.g., FTP and Telnet) and for several active queue management schemes.

5.2 Active Queue Management

Initially, Active Queue Management (AQM) were designed to minimize the the impact of network traffic and high latency. The first congestion scheme that implemented AQM was introduced by Floyd is called *Random Early Detection* (RED) algorithm [173]. Later on, this issue has been extensively addressed.

According to Drop Tail algorithm TCP protocol detects congestion only after a packet has been dropped from the queue. Keeping large queues full most of the time will significantly increase the delays. It is very important to have mechanisms that maintain a high overall throughput and at the same time maintain a low average queue size as possible. Furthermore, it is not necessary to keep queues completely empty all the time in order to maximize the throughput. Since this will result in under utilization of the link. Thus, the goal is to have high throughput and small queuing delay. Consequently, the queue length should be kept sufficiently small.

Additionally, drop tail considers two queue approaches that can be applied when the queue becomes full. They are: (i) *random drop on full* [200] and (ii) *drop front on full* [334]. Both approaches solve lock out problem, but neither of them solves full queue problem.

In order to fulfill the above goal, several AQM algorithms were proposed. The main objective of these algorithms is the early detection of network congestion and dropping packets before the network throughput is influenced. Nonetheless, this influence depends on the quality of service to be delivered by the network.

AQM algorithms utilize various metrics in order to estimate the network congestion. Thus, some AQM algorithms involve congestion metrics like: queue length, load, both queue length and load, loss rate, etc. Other algorithms use those metrics along with flow information to analyze and control congestion more accurately.

Broadly, AQM algorithms are classified into two classes: *reactive* and *proactive* [312]. Following, a brief description of these classes:

5.2.1 Reactive AQM Algorithms

The goal of *reactive AQM algorithms* is congestion avoidance by active early detection of and reaction to congestion. The status of current congestion determines the decisions on the actions to be taken. According to the criteria on which the decision is made whether to drop packets from the queue or not, whenever a congestion is occurred, four strategies of queue management can be identified:

Average queue length-based algorithms

In this class of reactive AQM algorithms, the dropping decision is based on the observed average queue length. Algorithms in this class can be divided further based on whether or not the algorithm provides a fairness on distribution of the available bandwidth over the active data flows. RED [173] is the basis of all algorithms in this class of AQM. Other algorithms of this class, for example, ARED [329], FRED [241], CHOKe [282].

Packet loss and link utilization-based algorithms

The algorithms of this class use packet loss and link utilization for performing active queue management rather than using the instantaneous or average queue lengths. A single probability p is maintained, which is used to drop packets when they are queued. In case such as buffer overflow the queue is continually dropping, p is incremented which increasing the rate of sending back congestion notification. In the other hand, if the queue becomes empty or if the link is idle, p is decremented. Member algorithms of this class are, for example, BLUE and SFB [169], SFED [221], YELLOW [245], etc.

Class-based algorithms

These algorithms treat the incoming packet depending on its class. In general, there are many possible class definitions, but in practice, it is common the incoming packets are categorized based on their transport protocol (i.e., TCP or UDP). Usually, for every non-TCP class, a threshold is defined, to limit the maximum number of packets a certain class can have in the queue. The Class-Based Threshold (CBT) [284] and Dynamic CBT (DCBT) [117] are members of this class of AQM algorithms.

Control theory-based algorithms

The algorithms of this class are based on the classical control theory techniques. These AQM algoritms attempt to maintain the instantaneous queue length as close as possible to a desired value (reference input). PI-controller [202] and Adaptive Virtual Queue (AVQ) [225] are examples of this class of AQM.

Hybrid-based algorithms

They make use of the benefits of each class to improve the drawbacks of another class group. Examples of this class of algorithms are: Random Early Marking (REM) [231], Stabilized Virtual Buffer (SVB) [151] and RaQ [330].

5.2.2 Proactive AQM Algorithms

The congestion prevention is the aim of *proactive AQM algorithms* which use intelligent and proactive dropping of packets. The expected congestion determines the decisions on the being taken actions. Examples of this type are: GREEN [116] and PAQM [311].

AQM algorithms provide an early detection of network congestion. In addition, several AQM algorithms perform flow-based filtering and throttle flows which experience abnormal behavior, for example:

- SRED [281] identifies flows that are taking more than their fair share of bandwidth and allocates a fair share of bandwidth for all flows;
- RRED [356] detects DoS flows and throttle those detected attack flows (RRED used to defend against low rate DoS); and
- Lin et al. [240] proposed a priority queue-based scheme to defend a gains DDoS attacks and compared the proposed scheme with Droptail and RED schemes.

Furthermore, Albiz in [26] considered AQM algorithms as a tolerance mechanism in his taxonomy of defense mechanisms of DoS attacks.

In Table 5.1, we give a summary of the important features of popular AQM algorithms.

5.3 Attack Models

Denial of Service Attacks (DoS) have been introduced earlier in chapter 4. In general, there are two classes of DoS attacks: *Flooding Attacks* and *Corruption Attacks*. In flooding attacks, an attacker sends superfluous requests at a high rate to overload the victim's resources (such as queues and CPU). While, in corruption attacks an attacker exploits a vulnerability in the victim using a few crafted requests in order to deny a service, for example, an attacker might alter the configuration information to prevent the use of a computer or network. Our focus in this chapter 5 is flooding DoS attacks.

Flooding DoS attacks can be classified as: *point-to-point* or *distributed* [153]. Examples of flooding DoS attacks are TCP SYN, UDP flooding, ICMP flooding and Smurf attacks.

Distributed DoS (DDoS) attacks combine point-to-point DoS attacks with distributed and coordinated control. The typical DDoS network model is shown in Figure 5.1, with one or more *attackers* controlling *handlers*, with each handler controls hordes of *agents* that execute the commands relayed to them. The goal of the two descending layers (i.e., handlers and agents) in this structure is twofold; increasing the rate of traffic and hiding the attacker from view. In case of some of the agents have been identified and have been taken off-line, the attacker has the ability to monitor the attack status and create new agents [153].

In order to have a reasonable simulation environment, an attention should be payed for choosing the attack simulation parameters to generate a sufficient attack traffic versus legitimate traffic. For the purpose of this study, we simulate a DDoS attack using UDP flooding which is generated by several attack nodes. The attack nodes are set to send attack traffic at a point of time during the simulation time, at the same time with the same rate (attack rate) and for the same duration (attack duration).

5.4 Network Topology

We use a dumbbell topology as shown in figure 5.2[1]. There are 30 legitimate nodes and 20 attack nodes, each of them is connected to the router (R_1) with a link that has $10Mbps/2ms$ bandwidth. The bottleneck link connects R_1 and R_2 and it has $5Mbps/10ms$ bandwidth. We use two NS-2 modules to generate legitimate traffic namely: *Application/FTP* and *Application/Telnet*. Attack traffic is generated using UDP flood.

5.5 DoS Impact Measurement

The *availability* of resources is the key issue of any system, which refers to a spectrum of service quality. Generally, the impact of DoS attacks is reducing the availability

[1] Adapted from [357]

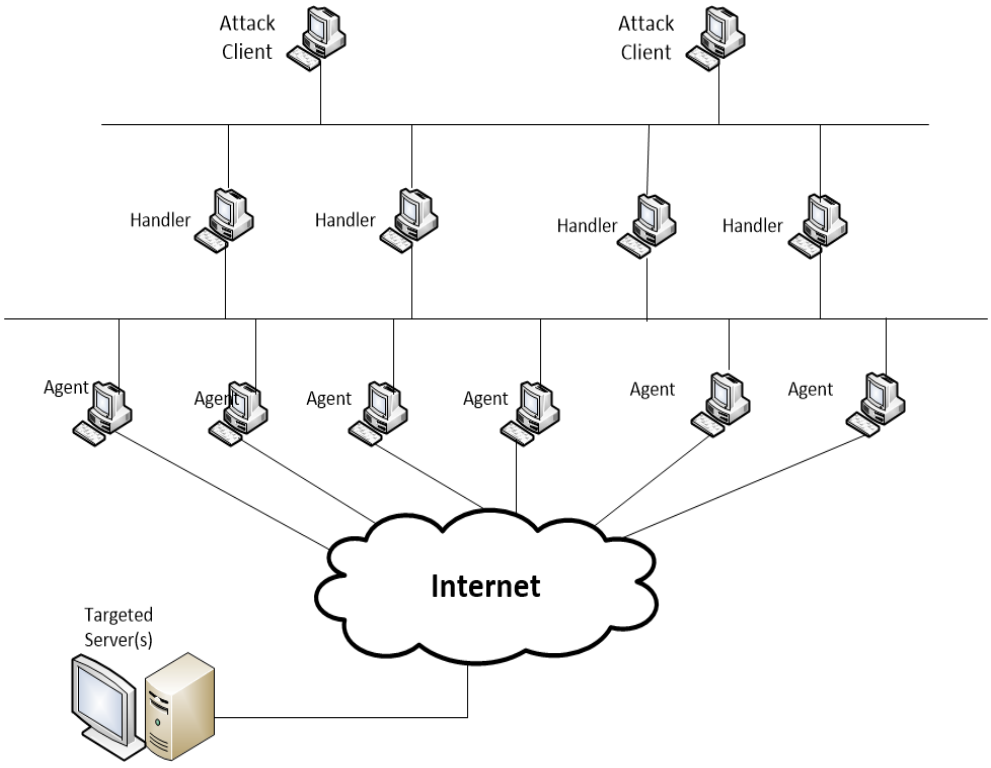


Figure 5.1: Typical DDoS network

(or Quality of Service (QoS)) of the system. Although quality of service metrics vary depending on the system or the service, they have been used to benchmark the availability. For example, [88] suggested *performance*, *completeness*, *accuracy* and *capacity* as metrics to benchmark the availability of software RAID systems, while [109] used *response latency*, *request throughput* and *time to recover* metrics to quantify DoS impact on location services such as DNS.

However, the impact of one DoS attack type might be different from one system to another. That is to say, a system A is more resilient than system B for one type of DoS attacks but less resilient for another attack. Thus, it is hard to develop a general threat model that combines the required set of dimensions of different threat classes. This is due to the system-specificity of those dimensions.

For the purpose of this study, we investigate several traffic measurement metrics based on the QoS requirements of the applications used. Two applications have been used for generating legitimate traffic, namely: FTP and Telnet. Those metrics depend on throughput, packet loss and end-to-end delay measurements. *Throughput* is defined as the amount of data transfered from the source to the destination

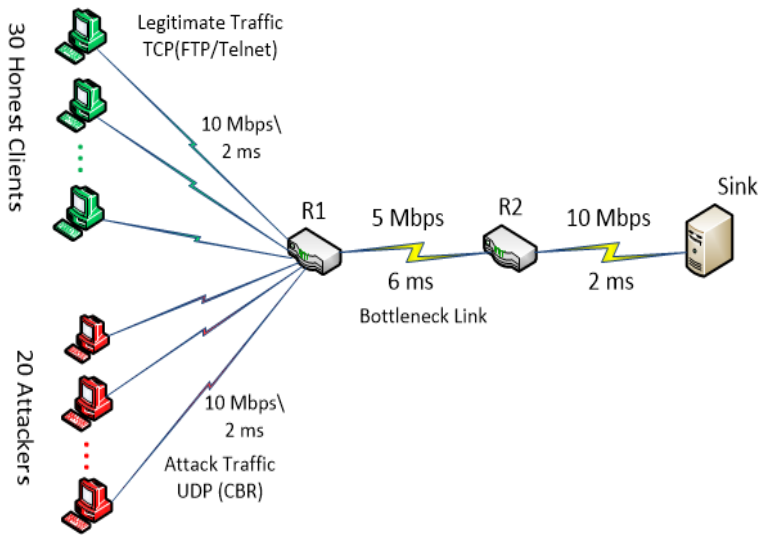


Figure 5.2: Experimental topology

per unit time. *Loss Rate* is defined by the number of packets or bytes lost due to the interaction of the legitimate traffic with the attack traffic or due to the defense operations. Here, we do not differentiate between the packet drops caused by the defense operation (such as early drops by AQM policy) and those caused by traffic interaction since they have the same impact on the end-user experience. Loss rate metric can be used to measure the presence of congestion in the network due to the flooding attacks. *End-to-end delay* is the interval between when a request is issued and when a complete response is received from the destination. A response time of less than 2 seconds and a delay of less than 250 millisecond are QoS requirement of telnet application to satisfy the human perception of users [110]. For FTP application, 2-5 seconds response time is recommended and approximately less than 1.2 seconds as delay [110][2].

The so-called investigated metrics as following:

Legitimate traffic throughput ratio during attack (LTR_{leg}) aims to quantify the impact of DoS attack on the network and to show the effectiveness of defense mechanism during the occurrence of the attack. It is a throughput-based metric and is

[2]The recommended delay must not exceed three times the expected delay experienced in the absence of the attack. In [110], it is assumed to be reasonably greater than the expected delay for a single web page which is < 400 ms.

calculated as following:

$$LTR_{leg} = \frac{Normal_{att}}{Normal}, \tag{5.1}$$

where;

$Normal_{att}$ is the average throughput rate of legitimate traffic during attack; and

$Normal$ is the average throughput rate of legitimate traffic when there is no attack.

The second metric is *Delivery ratio of legitimate packets during attack* (pdr_att_{leg}) which aims to quantify the QoS of the network during the attack. It is calculated according the following equation:

$$pdr\_att_{leg} = 1 - \frac{\text{no. of legitimate pkts dropped during attack}}{\text{no. of legitimate pkts sent during attack}} \tag{5.2}$$

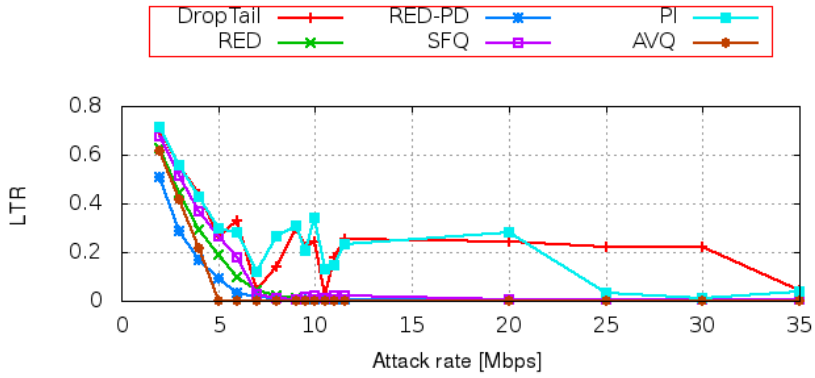
In this study, two traffic applications are considered, namely: (i) FTP traffic, and (ii) telnet traffic.

5.6 Experimental Results

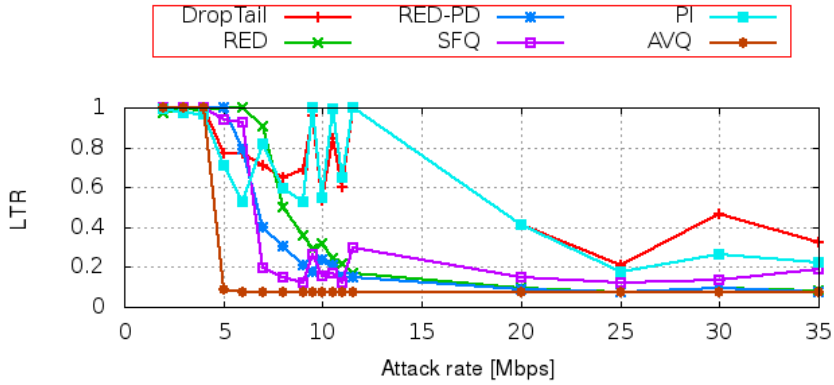
As stated above, those metrics vary from one application to another. In this experiment, two applications are used to generate the traffic of honest nodes namely: FTP and Telnet. The above metrics (see §5.5) are calculated and compared for both applications, where the server implements different AQM mechanisms. Though, the network settings are similar for both cases.

In this simulation, we consider only the UDP flooding attack which is frequently observed in the Internet and frequently used in literature. This attack can deny service in two ways: (1) by exhausting the bandwidth of the bottleneck link with a large amount of traffic, (2) by exhausting the CPU of the target or the CPU at a router leading to the target by sending high packet rate. In this study, we have generated the first attack type (a UDP bandwidth flood). There are 20 attack nodes that generate the required attack traffic. They start sending their traffic at the same time with the same bit rate and the same packet size. Packet size is set to (200 B) and bite rate varies for 0.1 Mbps to 1.75 Mbps for each attack node. The volume of the generated attack traffic varies from 2 Mbps to 35 Mbps which is able to congest the bottleneck link. This will lead legitimate packets to be dropped in the queue of the bottleneck link, thus the service is denied.

Figure 5.3 shows the Legitimate traffic throughput ratio during attack (LTR_{leg}) measures for both applications FTP, and Telnet. It is clear from this figure that the value of LTR_{leg} decreases by increasing the attack rate, which also varies depending on the application and the used defense. By comparing Figure 5.3a and Figure 5.3b, the effect of attack is less in the case of Telnet than that in case of FTP.



(a) FTP



(b) Telnet

Figure 5.3: Legitimate traffic throughput ratio (LTR_{leg}) during attack

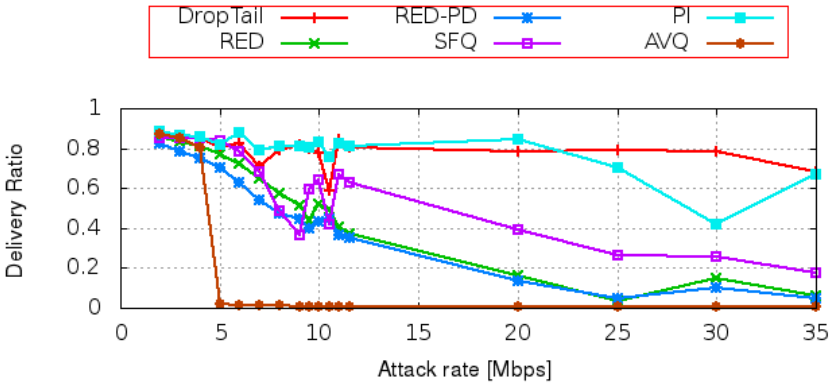
This is because Telnet generates much less traffic than FTP. Therefore, LTR_{leg} metrics is able to quantify the impact of attacks on the network and to evaluate the effectiveness of the used defense.

Figure 5.4 shows the measurements of the delivery ratio of legitimate packets during attack (pdr_att_{leg}) which aims to quantify the QoS that is maintained by the defense mechanism during the attack.

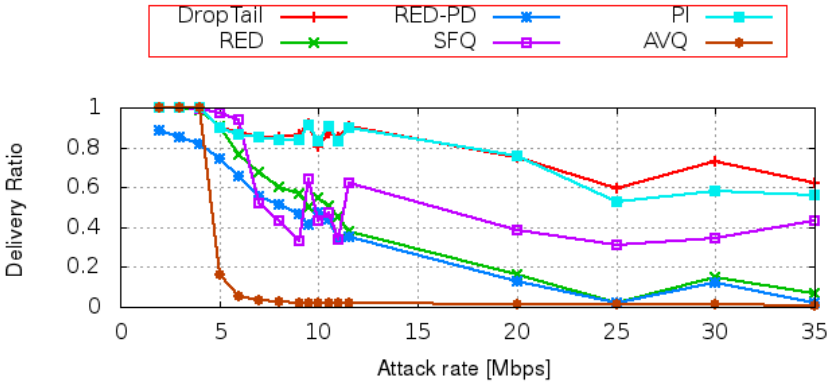
5.7 Summary

The used metrics are able to quantify the impact of DoS attack on the network bandwidth and to show the effectiveness of defense mechanisms. This emphasizes the usability of general network simulations in quantitative analysis of DoS attacks. However, the problem of developing effective testing methods of evaluating DoS attacks and their defenses is complicated. This is because of (i) the lack of complete attack profiles, and (ii) the dynamics of attack techniques.

In this direction, other possible future work could be: (1) creating realistic sim-



(a) FTP



(b) Telnet

Figure 5.4: Delivery ratio of legitimate traffic during attack (pdr_att_{leg})

ulation scenarios and traffic patterns, (2) there are several applications and services that are sensitive to the QoS experienced by users (application-level QoS). In this case, metrics that can capture the user experienced QoS are needed.

Finally, sharing experience, developing open realistic traffic repositories and tools, etc by the research community will help for better understanding of attacks and developing advanced testing methods.

Table 5.1: Summary of Some of AQM Algorithms [312]

Algorithm	Class	Important Features
RED	Average queue length-based QM	(1) It has been the basis of many other algorithms which attempted to rectify its cons. (2) It has been widely used wit TCP. (3) The main goal of RED algorithm is to provide congestion avoidance by controlling the average queue size.
ARED	Average queue length-based QM	(1) ARED uses heuristic approach in order to tune the RED's control parameters to enhance its robustness. (2) In particular, it attempts to control the rate of queue occupancy change, rather than controlling queue occupancy itself.
PD-RED	Control theory-based QM	(1) It is based on the *Proportional Derivation* (PD) control principle. (2) PD controller is used to stabilize the queue length. (3) Variation of queue length and the drop probability in PD-RED is smaller than in ARED. (4) It is sensitive to the short-lived and non-TCP traffics.
PI	Control theory-based QM	(1) It is based on a combination of two controller units: a *proportional controller* and *integral controller*. (2) The goal of the proportional controller is to steer the queue length to its desired length, while the integral controller is used to remove the steady-state regulation error in the queue length.
AVQ	Control theory-based QM	(1) AVQ is a rate based AQM. (2) It maintains virtual queue, whose capacity is less than the actual capacity of the router queue. (2) The virtual queue is updated when a packet is arrived in the actual queue. (3) When the virtual queue capacity overflows, a packet in actual queue is marked or dropped. (4) AVQ regulates the queue capacity instead of queue length.

Chapter 6

Conclusion and Future Work

Contents

6.1 Conclusion

Security requirements, such as *Confidentiality*, *Integrity*, *availability*, etc., must be achieved in order to exchange sensitive information over hostile environments such as Internet. To achieve those requirements is the aim of security protocols. However, the fast development of networking technologies and applications as well as the threat amount and techniques make it is necessary that new security protocols are continuously developing. Thus, the research on specification and verification of security protocols must continue to explore new methods in order to deal with the new emerging threats and to verify the requirements of the newly developed protocols.

The aim of this thesis is to use formal methods, in particular model checking techniques, to specify and verify different security protocols which provide different security requirements. The following is a summary of the thesis:

- A comprehensive survey of most important formal methods that have been developed for supporting the engineering of security protocols including: *security protocol specification and verification* and *security protocol design and synthesis* as well as security protocol repairing. Those formal techniques are based on different techniques such as model checking, modal logic and theorem proving, which depend on several mathematical concepts such as process algebra (e.g. π-calculus), timed automata, probabilistic models (e.g. Markov chains), predicate logic, etc.

- We presented an analysis of the security requirements of a security solution for authentication of inter-domain handover. Those requirements are authentication and secrecy which have been analyzed using the AVISPA toolkit. Our analysis identified a possible attack in the studied solution [34].

- We studied Denial of Service (DoS) attacks including: DoS attack characteristics, defense mechanisms and formal methods for analysis of DoS-resistance. Finally, we presented an method of quantifying the impact of DoS attacks on networks and evaluating the effectiveness of active DoS defense mechanisms through simulation using different performance metrics.

6.2 Outlook

In chapter (Chapter 4) we have presented the background knowledge of the Denial of Service (DoS) attacks and discussed different formal methods that have been used for modeling and analysis of DoS-resistance. One of the mentioned approaches was the cost-based framework [261] which is a well-defined quantitative formal technique. The major drawbacks of the cost-based method are: (1) it is a paper-pencil technique, and (2) the definition of costs is coarse and ad hoc. Our intention is to develop a formal approach to automate the cost-framework approach using probabilistic model checking. The costs of protocol actions are quantified using the benchmarks of cryptographic operations [147] similar as in [341].

The requirement of quantitative analysis and the challenges of specification of DoS-resistance requirement have been recognized previously [261, 260, 252, 36]. Since DoS-resistance requirements need to be expressed in degrees, a quantitative methods are needed. For example, it is reasonable to ask whether a proposed DoS-resistance mechanism is more effective than another. Another important issue of analyzing DoS-resistance requirements is the attacker model. Previously for analysis of secrecy properties, the popular *Dolev-Yao* [162] attacker model is used, which is not entirely suitable for analyzing DoS-resistance.

The actions of appropriate intruder model for verifying DoS-resistance requirements could be chosen as subset of the actions of Dolev-Yao with quantitative information to measure the available amount of resources to the attacker. Roughly speaking, the intruder model can implement several attack strategies such as replaying messages.

Initially, our goal is to to model the flooding attacks based on the cost-based framework. Thus, each intruder action is assigned a cost value that reflects the amount of resources that action requires when it is performed by the intruder. As a case study, the Just Fast Keying (JFK) protocol [31, 32, 9] which is a key establishment protocol with a DoS-resistance mechanism, is chosen (for more details about JFK protocol see Appendix C).

A further research in this direction emerging from two facts: (1) the evaluation of cryptographic functions varies depending on the processor speed, and (2) usually, protocols also perform some memory access operations. However, there are some research works that proposed puzzle constructions based on memory access functions (e.g. [16, 131]) to overcome the large speed differences between high-end processor and light weight devices in processing processor bounded puzzles. An interesting open question is to investigate the applicability of memory access functions as a

refinement for the costs of protocol operations.

6.3 Future Work

Based on our study of formal techniques in Chapter 2, it is clear that the main research of formal methods was targeting the authenticity and secrecy security requirements and there are many other requirements that have not been intensively explored such as *anonymity*, non-repudiation and availability.

Another general open research problem in the verification of security protocols is the timed analysis of security protocols. Because of the distribution nature of security protocols, time has a great effect in execution of security protocols. Time aspects in security protocols include *timestamps and time information* and *time flow of the protocol*. The formal such as lifetime of tokens, sessions, keys, and so on. The latter such as timeout and retransmission. It is known that time aspects of protocols can be exploited by an attacker to perform an attack. For example, low rate denial of service attacks can exploit the retransmission time-out mechanism of TCP protocol [228]. In this context, none of the devoted security protocol analysis tools provide analysis of time aspects except FDR/Casper that provides very limited features. However, there are other general purpose model checking tools (e.g. UPPAAL that was developed to analyze real-time systems) provide a rich representation of clocks. However, timed analysis of security protocols partially have been studied in few research works such as [136, 174, 211, 97].

The above mentioned problems are scheduled for later future work. On the other hand, there are some open research problems that have been uncovered in the current work, which are planned to be investigated in the near future. In the following, we list the most important problems:

- In chapter 3, we have analyzed the authenticity and secrecy of the authentication protocol of inter-domain handover. Another future study could be the analysis of the DoS-susceptibility of the protocol, in order to enhance the protocol DoS-resistance.

- In Chapter 5 we investigated several performance metrics in order to quantify the impact of DoS attack and evaluate the effectiveness of active DoS defense mechanisms. It is interesting to develop an analytical model to cooperate the simulation results with theoretical results.

Despite the massive research that have been done in the field of security protocol engineering, there is still an open issue of developing a methodology of design and verification of security protocols. This should provide a well-defined steps of security requirements acquisition, formal specifications, system definition and translation as well as analysis.

6.4 Final Remarks

The study of confidentiality and integrity requirement using formal methods has being developed for more than a decade and there is a deep understanding of how these requirements can be assured as well as vast array of formal techniques and tools that can be employed in their specification and verification. Nonetheless, new network applications (e.g. cloud computing) and protocols continuously are developed. As a consequence, new methods and tools need to be developed in order to deal with the new features. This research contributes in studying several formal techniques and tools for security protocols specification and verification as well as analyzing the security requirements of an authentication protocol for inter-domain handover.

The open infrastructure of Internet has a very important advantage that is the scalability. However, this design make it easy for attacker to launch sophisticated attacks that can deny the service of network. Nowadays, due to the intensive and sensitive use of communication networks and information systems, this highlights the importance of availability in general and DoS-resistance in particular. However, the verification of DoS-resistance requirements have not been explored using formal techniques comparing to the work that have been done for confidentiality and integrity analysis. The contribution of this research is investigating DoS problem, modeling and analyzing of DoS-resistance in authentication protocols and quantifying DoS impact and evaluating their defense mechanisms.

Appendix A

The Attack Trace: Explained in Chapter 3 §3.4.3

```
1
2  SUMMARY
3    UNSAFE
4
5  DETAILS
6    ATTACK_FOUND
7    TYPED_MODEL
8
9  PROTOCOL
10   /final_model_hash.if
11
12 GOAL
13   Authentication attack on (npaa,pac,npaa_pac_panakey,{AAA_Key_Int(10).
         n8(New_ns_id).n28(NPaC).n8(NnPAA)}_keygen)
14
15 BACKEND
16   CL-AtSe
17
18 STATISTICS
19   Analysed    : 823 states
20   Reachable   : 647 states
21   Translation: 0.07 seconds
22   Computation: 0.03 seconds
23
24 ATTACK TRACE
25  i -> (pac,9):   start
26  (pac,9) -> i:   n27(NewPaC_ID).npaa_id
27
28  i -> (npaa,4):   NewPaC_ID(7).npaa_id
29  (npaa,4) -> i:   npaa_id.cpaa_id
30
31  i -> (npaa,4):   npaa_id.caaa_pk
32  (npaa,4) -> i:   n8(New_ns_id).n8(NnPAA)
33
34  i -> (pac,9):   n8(New_ns_id).NnPAA(28)
35  (pac,9) -> i:   {n8(New_ns_id).n28(NPaC)}_pana_auth_key
36
37  i -> (npaa,4):   {n8(New_ns_id).n28(NPaC)}_pana_auth_key
38  (npaa,4) -> i:   {n8(New_ns_id).n8(NnPAA).npaa_id.npaa_pk}_(inv(naaa_pk
        ))
39
40  i -> (npaa,4):   {AAA_Key_Int(10)}_npaa_pk
```

```
41  (npaa,4) -> i:   {n8(New_ns_id).n10(New_Key_id).n10(Algo_id)}_({
        AAA_Key_Int(10).n8(New_ns_id).n28(NPaC).n8(NnPAA)}_keygen)
42
43  i -> (pac,9):   {n8(New_ns_id).n10(New_Key_id).n10(Algo_id)}_({
        AAA_Key_Int(10).n8(New_ns_id).n28(NPaC).n8(NnPAA)}_keygen)
44  (pac,9) -> i:   {n8(New_ns_id).n10(New_Key_id)}_({AAA_Key_Int(10).n8(
        New_ns_id).n28(NPaC).n8(NnPAA)}_keygen)
45
46  i -> (npaa,4):   {n8(New_ns_id).n10(New_Key_id)}_({AAA_Key_Int(10).n8(
        New_ns_id).n28(NPaC).n8(NnPAA)}_keygen)
47  (npaa,4) -> i:   ()
48                   & Secret({AAA_Key_Int(10).n8(New_ns_id).n28(NPaC).n8(
                         NnPAA)}_keygen,set_173);
49                   & Request(npaa,pac,npaa_pac_panakey,{AAA_Key_Int(10).
                         n8(New_ns_id).n28(NPaC).n8(NnPAA)}_keygen);
50                   & Add npaa to set_173;  Add pac to set_173;
```

Listing A.1: Attack Trace

Appendix B

Meadows's Cost-based Framework

B.1 Meadows's Cost-based Framework

The cost-based framework, was introduced in [261], provides a systematic approach for evaluating DoS resistance of a protocol by computing the cost incurred by the protocol participants in each step of the protocol. It views DoS resistance as a resource exhaustion problem. Meadows's cost-based framework is the first attempt at formalizing and analyzing DoS resistance in quantifiable manner.

Meadows's cost-based framework works by comparing the resource expenditure at both the initiator and the responder. This is done by calculating the cost of sequence actions of the protocol (costs are defined using *cost set*) comparing between the initiator and the responder. Meadows has used the cost sets; cheap, medium, and expensive to describe the resource commitment of protocol actions.

The following definitions are based on the definitions provided in the original paper [261].

B.1.1 Protocol Specifications

The framework uses an annotated style of the popular Alice-and-Bob notation that includes the associated steps of the generation, verification, and acceptance of messages.

Definition 1 (annotated Alice-and-Bob notation) *A message M of a protocol sent from A to B is written in the form*

$$A \rightarrow B : T_1, ..., T_k \parallel M \parallel O_1, ..., O_n,$$

where the T_i are the operations performed by A, and O_j are the operations performed by B. The sequence $T_1, ..., T_k$ represents the sequence of operations performed by A to produce M, while the sequence $O_1, ..., O_n$ represents the sequence of operations performed by B to process and verify M.

Definition 2 (protocol events) *Let*

$$L_i : A \rightarrow B : T_1, ..., T_k \parallel M \parallel O_1, ..., O_n,$$

be the i^{th} line in the protocol specification. An event X occurs in L_i if:

1. *X is one of T_i or O_j, or;*

2. X is 'A sends M to B' or 'B receives M from A'.

The events $T_i, ..., T_k$ and 'A sends M to B' are said to occur at A, while the events $O_1, ..., O_n$ and 'B receives M from A' are said to occur at B.

There are three types of events:

1. **Normal events** which include the send and receive events and can occur at either sender or receiver and always succeed and followed by the next events.

2. **Verification events** which only occur at the receiver side and may succeed or fail.

3. **Accept events** the event O_n is a reserved event called the accept event which indicates the completion of the process.

B.1.2 Cost Sets and Cost Functions

The participating costs in the protocol are derived by computing the costs of individual events and summing the costs for each of the steps in the protocol.

Definition 3 (cost set) *A cost set $\mathcal{C}$ is a monoid with operator $+$ and partial order $\leq$ such that $x \leq x + y$ and $y \leq x + y$, $forall x, y \in \mathcal{C}$.*

Definition 4 (event cost function) *The event cost function δ maps events defined in the protocol specification to a cost set $\mathcal{C}$ and is 0 on accept events.*

Definition 5 (message processing cost function) *The message processing cost function δ' is associated with event cost function δ and is defined on verification events $\{V_i\} \subset \{O_j\}$. If the line*

$$A \rightarrow B : T_1, ..., T_k \parallel M \parallel O_1, ..., O_n$$

appears in the protocol specification, then for each verification event $V_i = O_j$:

$$\delta'(V_i) = \delta(O_1) + ... + \delta(O_j).$$

Definition 6 (protocol engagement cost function) *The protocol engagement cost function Δ is defined on accept event O_n such that $\Delta(O_n)$ is the sum of all costs of all events occurring at the receiver up to O_n, plus the costs associated with any immediate response (i.e. the costs of $\delta(T_1) + ... + \delta(T_k)$ of the next line in the protocol specification).*

B.1.3 Intruder Cost Functions

The intruder is subject to independent set of cost functions that defines the costs of the actions that the intruder can perform in order to interfere with the protocol execution.

Definition 7 (intruder cost function) *Let $\mathcal{G}$ be the attacker cost set and $\mathcal{I}$ be the set of intruder actions. The function ϕ maps intruder actions to their costs in $\mathcal{G}$, is called* attacker event cost function. *The intruder cost function Φ is defined on a sequence of attacker actions as, $\Phi(\{i_1, ..., i_n\}) = \phi(i_1) + ... + \phi(i_n)$ for $i_k \in \mathcal{I}$.*

Definition 8 (fail-stop protocol) *An Alice-and-Bob specification of a cryptographic protocol is* fail-stop *if, whenever a message is interfered with, then no accept event desirably-after the receiving of that message will occur.*

Definition 9 (attack cost function) *The attacker cost function Θ maps events from a protocol specification $\mathcal{P}$ to cost set $\mathcal{C}$. $\mathcal{P}$ is fail-stop with respect to Θ if for every event $E \in \mathcal{P}$, if an attacker interferes with a message that should arrive before E, then no events that should occur after E will occur unless the cost to the attacker is at least $\Theta(E)$.*

Definition 10 (tolerance relation) *Let $\mathcal{C}$ and $\mathcal{G}$ be the responder and attacker cost sets respectively. A* tolerance relation *$\mathcal{T}$ is the subset of $\mathcal{C} \times \mathcal{G}$ that consists of all pairs (c, g) such that the responder will expend cost c only if the attacker will expend resources of at least cost g. A tuple (c', g') is said to be within the tolerance relation if there exists $(c, g) \in \mathcal{T}$, such that $c' \leq c$ and $g' \geq g$.*

B.1.4 General Steps of Assessing a Protocol's DoS Resistance

Meadows described a general procedure for evaluating a protocol's susceptibility to DoS, that includes the following steps:

1. determine the intruder capabilities and the cost functions;
2. decide on the tolerance relation $\mathcal{T}$;
3. determine the attack cost function Θ for each step of the protocol; and
4. for each Θ in step 3, verify that:
   i. if the event E_1 is immediately preceding a verification event E_2, then $(\delta'(E_1), \Theta(E_2)) \in \mathcal{T}$; and
   ii. if E is an accept event, then $(\Delta(E), \Theta(E)) \in \mathcal{T}$.

Having all the events of the protocol translated into the computational costs of the protocol parties, a comparison of all events of the protocol run between the sender and responder can be performed. If the final cost of the attacker is great

enough in comparison with the responder cost of engagement in the protocol up to accept event, the protocol is DoS-resistant. Otherwise, the protocol is susceptible to DoS attacks.

Table B.1 lists the costs of some protocol actions and Table B.2 lists the cost of attacker actions during the protocol run.

Action	Cost
`precompute`: precomputing of an operation	cheap
`generate_nonce`: generating a nonce	cheap
`exp`: compute an exponential function	expensive
`compute_HASH`: compute hash function	cheap to expensive (depending on the number of bits to be computed)
`verify_HASH`: verifying hash value	cheap to medium
`Symmetrical Encryption`: which includes	
(1) `compute_keys`	medium
(2) `encrypt`	medium
(3) `decrypt`	medium
`Asymmetrical Encryption`: which includes	
(1) `sign`	medium
(2) `verify_sign`	expensive

Table B.1: Costs of Protocol Actions

In this framework the cost sets $\mathcal{C}$ and the attacker cost set $\mathcal{G}$ are defined as:

$$0 < cheap < medium < expensive$$

In order to assess a protocol susceptibility against DoS attack, the tolerance relation $\mathcal{T}$ (Definition 10) is examined. That is to say, a protocol is DoS-resistant if the subset $(\mathcal{C}, \mathcal{G})$ belongs to acceptable tuples including[1]:

$$\mathcal{T} = \left\{ \begin{array}{l} (cheap, cheap); (cheap, medium); \\ (cheap, expensive); (medium, cheap); \\ (medium, medium); (medium, expensive); \\ (expensive, expensive) \end{array} \right\}$$

[1] The protocol analyst may include other tuples.

Action	Cost
1) Sending a legitimate message	= the cost of computing the message
2) Forging a return address	= cheap
3) Creating a new message out of old one	= cost of deconstructing the old message + cost of creating the new one
4) Disabling of a legitimate user	= medium
5) Reading messages	= medium
6) Breaking cryptosystem	= maximal
7) Substituting bogus messages for genuine ones in real time (man-in-the-middle attack)	= very expensive
8) Inducing a user to initiate communication with a bogus, disabled, or dishonest user	= expensive to very expensive

Table B.2: Attacker Costs [261]

Appendix C

Just Fast Keying Protocol

C.0.5 Just Fast Keying (JFK) Protocol

The Just Fast Keying (JFK) protocol has been discussed in several research papers and Internet Drafts [31, 32, 9]. In addition, JFK has been studied further by many researchers [15, 323, 149].

JFK is a key establishment protocol intended to be used at the beginning of the IPSec session to set up the security association[1]. The aim of JFK protocol to overcome the shortcomings of IKE protocol such as; its inefficiency and complexity as well as vulnerability to DoS attacks. JFK consists only of four messages and it provides several features like protecting the identity of participants, avoiding complex negotiations, efficiency, and resistance to memory and computational DoS attacks.

The designers of the protocol proposed two alternative protocols: first one is called JFKi which provides identity protection for the initiator (or client), and the second one is called JFKr which provides identity protection for the responder (or server). Both variants of JFK implement several defense techniques to counterbalance computational and memory exhaustion attacks as well as to gradually authenticate the initiators. The JFK protocol defers performing computationally expensive operations until the initiator has revealed its identity (IP address) and the responder is confident that it is round trip communication with the initiator. The JFKi variant of the protocol, which implements identity protection for the initiator against active attackers is shown in Figure C.1.

C.0.5.1 Protocol Description

As the responder requires to compute Diffie-Hellman (DH) exponential (g^r) which is an expensive operation, the protocol designers allows this value to be used for different sessions. Thus, the responder periodically chooses a DH exponential and generates a signature over this value and information on the groups it supports. The reuse of g^r with multiple initiators reduces the computational expenditure at the expense of perfect forward security (PFS).

By receiving the first message, the responder (R) remains stateless and constructs the authenticator (cookie) which will be sent to the initiator (I) in the second message. In message three, the initiator returns the authenticator (cookie) along with the source of its hashed nonce N_I provided in the first message. This

[1] A security association (SA) is A set of security parameters including session keys, initialization vectors or digital certificates.

$$
\begin{aligned}
Msg1 : I \rightarrow R :& N'_I, g^i, ID'_R; \\
& \text{where } N'_I = H(N_I) \\
Msg2 : R \rightarrow I :& N'_I, N_R, g^r, grpinfo_R, ID_R, Cookie, sig_{R1}; \text{ where} \\
& Cookie = H_{HK_R}(g^r, N_R, N'_I, IP_I) \\
& sig_{R1} = S_R[g^r, groupinfo_R] \\
Msg3 : I \rightarrow R :& N_I, N_R, g^i, g^r, Cookie, E_1; \text{ where} \\
& K_e = H_{g^{ir}}(N'_I, N_R, '1') \\
& K_a = H_{g^{ir}}(N'_I, N_R, '2') \\
& K_s = H_{g^{ir}}(N'_I, N_R, '0') \\
& sig_I = S_I[N'_I, N_R, g^i, g^r, ID_R, sa_I] \\
& E1 = \{ID_I, sa_I, sig_I\}^{K_e}_{K_a} \\
Msg4 : R \rightarrow I :& E2; \text{ where} \\
& N'_I = H(N_I) \\
& Cookie \stackrel{?}{=} H_{HK_R}(g^r, N_R, N'_I, IP_I) \\
& K_e = H_{g^{ir}}(N'_I, N_R, '1') \\
& K_a = H_{g^{ir}}(N'_I, N_R, '2') \\
& K_s = H_{g^{ir}}(N'_I, N_R, '0') \\
& \text{verify and decrypt } E1 \\
& \text{verify } sig_I \\
& sig_{R2} = S_R[N'_I, N_R, g^i, g^r, ID_I, sa_I, sa_R] \\
& E2 = \{sig_{R2}, sa_R\}^{K_e}_{K_a}
\end{aligned}
$$

Figure C.1: JFKi Protocol[32]

allows the responder to cheaply verify that the sender of the third message is also the sender of the first message. After the verification of the authenticator (cookie), which includes the verification of the initiator's nonce preimage, a valid authenticator indicates that a round-trip has been completed. The responder commits to the protocol by verifying the initiator's signature (included in the third message) and generating session keys.

C.0.5.2 JFK's DoS Resistance

One of the design goals of the JFK protocol is to resist memory and computational DoS attacks. Following, a brief description of the JFK's DoS-resistance techniques.

Resistance Against Memory Exhaustion Attacks

After receiving the initiator's first message, the responder remains stateless and weakly authenticates the initiator by generating an authenticator (a cryptographic cookie) which is sent in second message to the initiator and returned back by the initiator in the third message.

$$Cookie = H_{HK_R}(g^r, N_R, N'_I, IP_I)$$

The key HK_R is MAC key known only to the responder and it is time variant that limits the period of time that the cookie is valid.

Resistance Against Computational DoS Attacks

The protocol allows the reuse of Deffie-Hellman exponential to reduce computational expenditure at the responder. Allowing the reuse of DH exponential by the responder reduces the cost of generating the signature in message two. However, the JFK protocol does not provide a mechanism to increase computational expenditure at the initiator. This can make the responder susceptible to computational DoS attacks, when there are some initiators willing to reveal their identity. It is possible, an attacker starts the protocol with the responder with a legitimate message 1, then cheaply fabricate a bogus message 3. Thus, the responder would have engaged in performing an expensive modular exponentiation before it recognizes that the received message was bogus. Smith et al in [323] explored the use of client puzzle to JFK protocol as a proof of work to increase the computational expenditure of an initiator. The verification of puzzle solution only requires a single hash operation.

Gradual Authentication

In the first message, the initiator sends only a hashed value of his nonce (N'_I), while in the third message, he releases the source (N_I) of that hashed value. This allows the responder to validate that third message is from the same initiator. Furthermore, the initiator has to derive the DH key (g^{ir}) and the encryption and authentication keys used to protect the third message.

By receiving third message, the responder gradually authenticate the initiator by performing several checks before performing the signature verification to strongly authenticate the initiator. The responder, at first, validates the nonce (N_I) and then ensures the reachability of the initiator at the given address in first message by validating the cookie. After the successful completion of these checks, the responder then decrypts and verifies the contents of third message and finally verifies the initiator's signature.

Performing an expensive cryptographic operation in third message by the initiator provides the responder assurance of the initiator's willingness to proceed in the protocol and commit computational resources. However, this approach requires also the responder to use an equivalent computational resources as the initiator, since he has to decrypt and verify the encryption and the MAC of third message. There

is no counterbalancing between the effort of the initiator and the responder like in the case of client puzzles. However, the failure by the responder to decrypt or verify the MAC of third message allows him to detect possible attack prior performing an expensive signature verification.

C.1 Analysis of JFK

According to the Meadows's cost-based framework, the protocol events are defined and assigned costs. The main events of JFK protocol curried out by each of the initiator and responder for each message are summarized in the Table C.1. In this table, we associate each event a cost based on the cost of the original cost-based framework [261] (which either *cheap*, *medium* or *expensive*).

In Table C.2 the JFK protocol in the cost-based framework notation is presented.

Message	Initiator (I) Events	Responder (R) Events
$I \rightarrow R$: $\{Msg.1\}$	Compute nonce N_I: `compute_nonce(`N_I`) = cheap`	Verify group of g^i: `verifygroup(`g^i`) = medium`
	Hash of N_I: `hash(`N_I`) = cheap`	
	Generate DH exp. (g^i): `generateHD(`g^i`) = expensive`	
$R \rightarrow I$: $\{Msg.2\}$	Verify Sig_R1: `verifySig(`sig_R1`) = expensive`	Compute nonce N_R: `compute_nonce(`N_R`) = cheap`
		Generate MAC: `Cookie = generateMAC(`$K_R, \{g^r, N_R, N'_I, IP_I\}$`) = medium`
		Generate signature sig_R1: `generateSig(`sig_R1`) = expensive`
$I \rightarrow R$: $\{Msg.3\}$	Generate DH exp. g^{ir}: `generateHD(`g^{ir}`) = expensive`	Hash of N_I: `hash(`N_I`) = cheap`
		Verify cookie: `verify(Cookie = generateMAC(`$K_R, \{g^r, N_R, N'_I, IP_I\}$`)) = medium`
	Compute keys K_e and K_a: `K1 = computeKey(`g^{ir}, N'_I, N_R`) = medium`	Generate DH exp. g^{ir}: `generateHD(`g^{ir}`) = expensive`
	Generate signature sig_I: `generateSig(`sig_I`) = expensive`	Compute keys K_e and K_a: `K2 = computekey(`g^{ir}, N'_I, N_R`) = medium`
	Encrypt ID_I, sig_I, sa: `E1 = encrypt(`$K1, \{ID_I, sig_I, sa_I\}$`) = medium`	Verify I's MAC of $(K1, E1)$: `verify(generateMAC(`$K1, E1$`)) = medium`
	Generate MAC: `generateMAC(`$K1, E1$`) = medium`	Decrypt $E1$: `decrypt(`$K2, E1$`) = medium`
		Verify signature sig_I: `verifySig(`sig_I`) = expensive`
$R \rightarrow I$: $\{Msg.4\}$	Verify R's MAC of $(K2, E2)$: `verify(generateMAC(`$K2, E2$`) = medium`	Generate signature sig_R2: sig_R2 = `generateSig(`$N'_I, N_R, g^i, g^r, ID_I, sa_I, sa_R$`) = expensive`
	Decrypt $E2$: `decrypt(`$K1, E2$`) = medium`	Encrypt $K2, sig_R2, sa_R$: `E2 = encrypt(`$K2, \{sig_R2, sa_R\}$`) = medium`
	Verify signature sig_R2: `verifySig(`sig_R2`) = expensive`	Generate MAC: `generateMAC(`$K2, E2$`) = medium`

Table C.1: JFK protocol events and their associated costs according to cost-based framework

L1. $I \rightarrow R$: $\texttt{compute\_nonce}_1(N_I)$, $N'_I = \texttt{hash}(N_I)$, $\texttt{generateHD}_1(g^i)$ $||N'_I, g^i, ID'_R||$ $\texttt{verifygroup}(g^i)$, $\texttt{accept}_1$

L2. $R \rightarrow I$: $\texttt{compute\_nonce}_2(N_R)$, $\texttt{Cookie} = \texttt{generateMAC}_1(K_R, g^r, N_R, N'I, IP_I)$, $sig_{R1} = \texttt{generateSig}_1(g^r, \texttt{groupinfo}_R)$ $||$ $N'_I, N_R, groupinfo_R, ID_R, Cookie, sig_{R1}$ $||$ $\texttt{verifySig}_1$, $\texttt{accept}_2$

L3. $I \rightarrow R$: $\texttt{generateHD}_2(g^{ir})$, $K_1 = \texttt{computeKey}_1(g^{ir}, N'_I, N_R)$, $sig_I = \texttt{generateSig}_2(N'_I, N_R, g^i, g^r, ID_R, sa_I)$, $E_1 = \texttt{encrypt}_1(K_1, \{ID_I, sig_I.sa_I\})$, $C_1 = \texttt{generateMAC}_2(K_I, E_1)$ $||N_I, N_R, g^i, g^r, Cookie, E_1, C_1||$ $N'_I = \texttt{hash}_2(N_I)$, $\texttt{verify}_1(\texttt{Cookie} = \texttt{generateMAC}_3(K_R, \{g^r, N_R, N'_I, IP_I\})$, $\texttt{generateHD}_2(g^{ir})$, $K_2 = \texttt{computeKey}_2(g^{ir}, N'_I, N_R)$, $\texttt{verify}_2(C_1 = \texttt{generateMAC}_4(K_1, E_1)$, $\texttt{decrypt}_1(K_2, E_1)$, $\texttt{verifySig}_2(sig_I)$, $\texttt{accept}_3$

L4. $R \rightarrow I$: $sig_{R2} = \texttt{generateSig}_3(N'_I, N_R, g^i, g^r, ID_I, sa_I, sa_R)$, $E_2 = \texttt{encrypt}_2(K_2, \{sig_{R2}, sa_R\})$, $C_2 = \texttt{generateMAC}_4(K_2, E_2)$ $||E_2, C_2||$ $\texttt{verify}_3(C_2 = \texttt{generateMAC}_6(K_2, E_2)$, $\texttt{decrypt}_2(E_2)$, $\texttt{verifySig}_3(sig_{R2})$, $\texttt{accept}_4$

Table C.2: JFK protocol in the cost-based framework notation

Bibliography

[1] http://www.joe-ks.com/archives_aug2003/BikeLocks.jpg. (Cited on pages 1 and 13.)

[2] http://en.wikipedia.org/wiki/File:EnigmaMachineLabeled.jpg. (Cited on pages 1 and 14.)

[3] AVISPA: automated validation of internet security protocols and applications (2003) FET Open Project IST-2001-39252.
www.avispa-project.org. (Cited on page 31.)

[4] http://www.mitre.org/tech/strands/. (Cited on page 34.)

[5] –, âAVISPA Library â, http://www.avispa-project.org/library/. (Cited on page 72.)

[6] "VoIP Security and Privacy Threat Taxonomy-Public Release 1.0", VOIPSA, October 2005. (Cited on page 81.)

[7] Available on: http://insecure.org/sploits/ping-o-death.html. (Cited on page 81.)

[8] Bysin, Knight.c sourcecode, PacketStormSecurity.nl July 11, 2001, Available from <http://packetstormsecurity.nl/distributed/knight.c>. (Cited on page 83.)

[9] W. Aiello, S. Bellovin, M. Blaze, R. Canetti, J. Ionnidis, A. Keromytis, and O. Reingold. Just fast keying (JFK). IETF Internet Draft draft-ietf-ipsec-jfk-04.txt, July 2002. (Cited on pages 118 and 129.)

[10] CERT Advisory Ca-1998-01: Smurf Ip Denial-of-Service Attacks. Available online at: http://www.cert.org/advisories/CA-1998-01.html. (Cited on page 80.)

[11] Y. Boichut Y. Chevalier L. Compagna J. Cuellar P. Hankes Drielsma P.C. Heám O. Kouchnarenko J. Mantovani S. Mödersheim D. von Oheimb M. Rusinowitch J. Santiago M. Turuani L. Viganò L. Vigneron. A. Armando, D. Basin. The AVISPA tool for the automated validation of internet security protocols and applications, 2005. in proceedings of CAV 2005, Computer Aided Verification, LNCS 3576, Springer Verlag. http://avispa-project.org/. (Cited on pages 6 and 20.)

[12] Martín Abadi. Secrecy by typing in security protocols. *J. ACM*, 46(5):749–786, 1999. (Cited on page 38.)

[13] Martín Abadi and Bruno Blanchet. Computer-assisted verification of a protocol for certified email. *Sci. Comput. Program.*, 58(1-2):3–27, 2005. (Cited on page 39.)

[14] Martín Abadi, Bruno Blanchet, and Cédric Fournet. Just fast keying in the pi calculus. *ACM Trans. Inf. Syst. Secur.*, 10(3):9, 2007. (Cited on page 39.)

[15] Martín Abadi, Bruno Blanchet, and Cédric Fournet. Just fast keying in the pi calculus. *ACM Trans. Inf. Syst. Secur.*, 10(3):9, 2007. (Cited on page 129.)

[16] Martin Abadi, Mike Burrows, Mark Manasse, and Ted Wobber. Moderately hard, memory-bound functions. *ACM Trans. Internet Technol.*, 5(2):299–327, 2005. (Cited on page 118.)

[17] Martín Abadi and Cédric Fournet. Mobile values, new names, and secure communication. *SIGPLAN Not.*, 36(3):104–115, 2001. (Cited on page 39.)

[18] Martín Abadi and Andrew D. Gordon. A calculus for cryptographic protocols: The spi calculus. In *Fourth ACM Conference on Computer and Communications Security*, pages 36–47. ACM Press, 1997. (Cited on page 37.)

[19] Martín Abadi and Andrew D. Gordon. A bisimulation method for cryptographic protocols. *Nordic J. of Computing*, 5(4):267–303, 1998. (Cited on page 38.)

[20] Martín Abadi and Roger Needham. Prudent engineering practice for cryptographic protocols. *IEEE Trans. Softw. Eng.*, 22(1):6–15, 1996. (Cited on pages 12 and 45.)

[21] Martín Abadi and Phillip Rogaway. Reconciling two views of cryptography (the computational soundness of formal encryption). In *Proceedings of the International Conference IFIP on Theoretical Computer Science, Exploring New Frontiers of Theoretical Informatics*, TCS '00, pages 3–22, London, UK, 2000. Springer-Verlag. (Cited on page 21.)

[22] Martín Abadi and Phillip Rogaway. Reconciling two views of cryptography (the computational soundness of formal encryption). In *TCS '00: Proceedings of the International Conference IFIP on Theoretical Computer Science, Exploring New Frontiers of Theoretical Informatics*, pages 3–22, London, UK, 2000. Springer-Verlag. (Cited on pages 21 and 22.)

[23] MartÃn Abadi, Bruno Blanchet, and D Epartement D'informatique. Analyzing security protocols with secrecy types and logic programs. In *Journal of the ACM*, pages 33–44. ACM Press, 2002. (Cited on page 39.)

[24] H. Abdelnur, T. Avanesov, M. Rusinowitch, and R. State. Abusing SIP authentication. In *Information Assurance and Security, 2008. ISIAS '08. Fourth International Conference on*, pages 237 –242, sept. 2008. (Cited on page 72.)

[25] Mehmud Abliz. Internet denial of service attacks and defense mechanisms. University of Pittsburgh Technical Report, No. TR-11-178, March 2011, Pages 1-50. (Cited on page 82.)

[26] Mehmud Abliz. Internet denial of service attacks and defense mechanisms. University of Pittsburgh Technical Report, No. TR-11-178, March 2011. (Cited on page 108.)

[27] Mehmud Abliz and Taieb Znati. A guided tour puzzle for denial of service prevention. In *ACSAC '09: Proceedings of the 2009 Annual Computer Security Applications Conference*, pages 279–288, Washington, DC, USA, 2009. IEEE Computer Society. (Cited on pages 87 and 91.)

[28] Reynald Affeldt and Hubert Comon-Lundh. A note on first-order logic and security protocols., 2008. (Cited on page 41.)

[29] Gul Agha, Michael Greenwald, Carl A. Gunter, Sanjeev Khanna, Jose Meseguer, Koushik Sen, and Prasannaa Thati. Formal modeling and analysis of dos using probabilistic rewrite theories. In *In International Workshop on Foundations of Computer Security (FCSâ05) (Affiliated with LICSâ05*, 2005. (Cited on pages 98, 100 and 101.)

[30] Gul Agha, José Meseguer, and Koushik Sen. Pmaude: Rewrite-based specification language for probabilistic object systems. *Electron. Notes Theor. Comput. Sci.*, 153(2):213–239, 2006. (Cited on pages 100 and 101.)

[31] William Aiello, Steven M. Bellovin, Matt Blaze, Ran Canetti, John Ioannidis, Angelos D. Keromytis, and Omer Reingold. Just fast keying: Key agreement in a hostile internet. *ACM Trans. Inf. Syst. Secur.*, 7:242–273, May 2004. (Cited on pages 118 and 129.)

[32] William Aiello, Steven M. Bellovin, Matt Blaze, John Ioannidis, Omer Reingold, Ran Canetti, and Angelos D. Keromytis. Efficient, dos-resistant, secure key exchange for internet protocols. In *CCS '02: Proceedings of the 9th ACM conference on Computer and communications security*, pages 48–58, New York, NY, USA, 2002. ACM. (Cited on pages 1, 94, 96, 97, 118, 129 and 130.)

[33] M. Akbarzadeh and M. Azgomi. A framework for probabilistic model checking of security protocols using coloured stochastic activity networks and PDETool. In *Proc. 5th International Symposium on Telecommunications (IST'10)*, 2010. (Cited on page 37.)

[34] S. Al-shadly, O. Alfandi, H. Brosenne, and D. Hogrefe. Security verification of trust-based authentication handover in ip networks. In *Computer Communications and Networks (ICCCN), 2011 Proceedings of 20th International Conference on*, pages 1 –6, 31 2011-aug. 4 2011. (Cited on page 117.)

[35] O. Alfandi, H. Brosenne, C. Werner, and D. Hogrefe. Fast re-authentication for inter-domain handover using context transfer. In *International Conference on Information Networking (ICOIN)*, pages 1–5, 2008. (Cited on pages 54, 56, 58 and 72.)

[36] Musab AlTurki, José Meseguer, and Carl A. Gunter. Probabilistic modeling and analysis of dos protection for the asv protocol. *Electron. Notes Theor. Comput. Sci.*, 234:3–18, 2009. (Cited on pages 50, 96, 98, 100, 101 and 118.)

[37] Rajeev Alur and David L. Dill. Automata for modeling real-time systems. In *Proceedings of the 17th International Colloquium on Automata, Languages and Programming*, ICALP '90, pages 322–335, London, UK, 1990. Springer-Verlag. (Cited on pages 6 and 37.)

[38] Rajeev Alur and David L. Dill. A theory of timed automata. *Theor. Comput. Sci.*, 126:183–235, April 1994. (Cited on page 6.)

[39] Roberto M. Amadio, Roberto M. Amadio, Denis Lugiez, Denis Lugiez, Thème Réseaux Et Systèmes, Unité Inria, and Sophia Antipolis. On the reachability problem in cryptographic protocols. pages 380–394. Springer-Verlag, 2000. (Cited on page 39.)

[40] Tobias Amnell, Gerd Behrmann, Johan Bengtsson, Pedro R. D'Argenio, Alexandre David, Ansgar Fehnker, Thomas Hune, Bertrand Jeannet, Kim Guldstrand Larsen, M. Oliver Möller, Paul Pettersson, Carsten Weise, and Wang Yi. Uppaal - now, next, and future. In *Proceedings of the 4th Summer School on Modeling and Verification of Parallel Processes*, MOVEP '00, pages 99–124, London, UK, 2001. Springer-Verlag. (Cited on page 37.)

[41] Iván Arce and Elias Levy. An analysis of the slapper worm. *IEEE Security and Privacy*, 1(1):82–87, 2003. (Cited on page 19.)

[42] Myla Archer and Ben Di Vito. Developing user strategies in pvs: A tutorial. In *In: Proceedings of the First International Workshop on Design and Application of Strategies/Tactics in Higher Order Logics (STRATA*, pages 2003–212448, 2003. (Cited on page 47.)

[43] Alessandro Armando, Roberto Carbone, Jorge Cuellar, Llanos Tobarra, and Luca Compagna. Formal Analysis of SAML 2.0 Web Browser Single Sign-On: Breaking the SAML-based Single Sign-On for Google Apps. In *In Proceedings of FMSE 2008*. ACM Press, 2008. (Cited on page 72.)

[44] Alessandro Armando and Luca Compagna. Automatic sat-compilation of protocol insecurity problems via reduction to planning. In *FORTE '02: Proceedings of the 22nd IFIP WG 6.1 International Conference Houston on Formal Techniques for Networked and Distributed Systems*, pages 210–225, London, UK, 2002. Springer-Verlag. (Cited on page 33.)

[45] Alessandro Armando and Luca Compagna. Abstraction-driven sat-based analysis of security protocols. In *Theory and Applications of Satisfiability Testing*, volume Volume 2919/2004, pages 301–302, Berlin / Heidelberg, 2004. Springer-Verlag. (Cited on page 33.)

[46] Alessandro Armando and Luca Compagna. Satmc: A sat-based model checker for security protocols. In *JELIA*, pages 730–733, 2004. (Cited on page 33.)

[47] Alessandro Armando and Luca Compagna. An optimized intruder model for SAT-based model-checking of security protocols. *Electr. Notes Theor. Comput. Sci.*, 125(1):91–108, 2005. (Cited on page 33.)

[48] Alessandro Armando and Luca Compagna. Sat-based model-checking for security protocols analysis. *Int. J. Inf. Secur.*, 7(1):3–32, 2008. (Cited on page 56.)

[49] Alessandro Armando, Luca Compagna, and Pierre Ganty. Sat-based model-checking of security protocols using planning graph analysis. In *FME*, pages 875–893, 2003. (Cited on page 33.)

[50] Gavin Lowe August. An attack on the needham-schroeder public-key authentication protocol. *Information Processing Letters*, 56:131–133, 1995. (Cited on page 5.)

[51] Tuomas Aura, Pekka Nikander, and Jussipekka Leiwo. Dos-resistant authentication with client puzzles. In *Revised Papers from the 8th International Workshop on Security Protocols*, pages 170–177, London, UK, 2001. Springer-Verlag. (Cited on pages 86, 87, 88, 94 and 95.)

[52] Jason Barlow. TFN2K - An Analysis., 2000. Available on: http://www.securiteam.com/securitynews/5YP0G000FS.html. (Cited on pages 79 and 83.)

[53] Stylianos Basagiannis, Panagiotis Katsaros, and Andrew Pombortsis. Intrusion attack tactics for the model checking of e-commerce security guarantees. In *SAFECOMP*, pages 238–251, 2007. (Cited on pages 18 and 20.)

[54] Stylianos Basagiannis, Panagiotis Katsaros, Andrew Pombortsis, and Nikolaos Alexiou. A probabilistic attacker model for quantitative verification of dos security threats. *Computer Software and Applications Conference, Annual International*, 0:12–19, 2008. (Cited on page 98.)

[55] Stylianos Basagiannis, Panagiotis Katsaros, Andrew Pombortsis, and Nikolaos Alexiou. Probabilistic model checking for the quantification of dos security threats. *Computers and Security*, 28(6):450–465, 2009. (Cited on pages 98 and 100.)

[56] David Basin, Sebastian Moedersheim, and Luca Vigan. Ofmc: A symbolic model checker for security protocols. *Published online: 21 December 2004 - ©Springer-Verlag 2004*, 2004. (Cited on page 31.)

[57] David Basin, Sebastian Moedersheim, and Luca Vigan. Ofmc: A symbolic model checker for security protocols. *Published online: 21 December 2004 - ©Springer-Verlag 2004*, 2004. (Cited on pages 20 and 56.)

[58] David A. Basin. Lazy infinite-state analysis of security protocols. In *Proceedings of the International Exhibition and Congress on Secure Networking - CQRE (Secure) '99*, pages 30–42, London, UK, 1999. Springer-Verlag. (Cited on page 30.)

[59] Mathieu Baudet, Veronique Cortier, and Steve Kremer. Computationally sound implementations of equational theories against passive adversaries. In *Proc. 32nd International Colloquium on Automata, Languages and Programming (ICALPâ05), volume 3580 of LNCS*, pages 652–663. Springer, 2005. (Cited on page 22.)

[60] Gerd Behrmann, Alexandre David, Kim Guldstrand Larsen, Paul Pettersson, and Wang Yi. Developing uppaal over 15 years. *Softw., Pract. Exper.*, 41(2):133–142, 2011. (Cited on page 6.)

[61] Genge Bela and I. Ignat. Verifying the independence of security protocols. In *Intelligent Computer Communication and Processing, 2007 IEEE International Conference on*, pages 155–162, 2007. (Cited on page 45.)

[62] Giampaolo Bella. Inductive verification of cryptographic protocols, 2000. (Cited on page 43.)

[63] Giampaolo Bella. Inductive verification of smart card protocols. *J. Comput. Secur.*, 11(1):87–132, 2003. (Cited on page 43.)

[64] Giampaolo Bella and Lawrence C. Paulson. Using isabelle to prove properties of the kerberos authentication system. In *In DIMACS Workshop on Design and Formal Verification of Security Protocols*, 1997. (Cited on page 43.)

[65] Giampaolo Bella and Lawrence C Paulson. Kerberos version iv: Inductive analysis of the secrecy goals. In *Computer Security â ESORICS 98, LNCS 1485*, pages 361–375. Springer, 1998. (Cited on page 43.)

[66] John Bellardo and Stefan Savage. 802.11 denial-of-service attacks: real vulnerabilities and practical solutions. In *SSYM'03: Proceedings of the 12th conference on USENIX Security Symposium*, pages 2–2, Berkeley, CA, USA, 2003. USENIX Association. (Cited on page 81.)

[67] Mihir Bellare and Phillip Rogaway. Entity authentication and key distribution. In *Proceedings of the 13th annual international cryptology conference on Advances in cryptology*, pages 232–249, New York, NY, USA, 1994. Springer-Verlag New York, Inc. (Cited on page 46.)

[68] Johan Bengtsson, Kim Larsen, Fredrik Larsson, Paul Pettersson, and Wang Yi. Uppaal: a tool suite for automatic verification of real-time systems. In *Proceedings of the DIMACS/SYCON workshop on Hybrid systems III : verification and control: verification and control*, pages 232–243, Secaucus, NJ, USA, 1996. Springer-Verlag New York, Inc. (Cited on page 6.)

[69] B. Berard, M. Bidoit, A. Finkel, F. Laroussinie, A. Petit, L. Petrucci, and P. Schnoebelen. *Systems and Software Verification: Model-Checking Techniques and Tools*. Springer Publishing Company, Incorporated, 1st edition, 2010. (Cited on page 7.)

[70] Karthikeyan Bhargavan, Cédric Fournet, Andrew D. Gordon, and Riccardo Pucella. Tulafale: A security tool for web services. In *In International Symposium on Formal Methods for Components and Objects (FMCO'03), volume 3188 of LNCS*, pages 197–222. Springer, 2004. (Cited on page 50.)

[71] Armin Biere, Ro Cimatti, Edmund Clarke, and Yunshan Zhu. Symbolic model checking without bdds. pages 193–207. Springer-Verlag, 1999. (Cited on pages 27 and 33.)

[72] Bruno Blanchet. An efficient cryptographic protocol verifier based on prolog rules. In *CSFW '01: Proceedings of the 14th IEEE workshop on Computer Security Foundations*, page 82, Washington, DC, USA, 2001. IEEE Computer Society. (Cited on page 36.)

[73] Bruno Blanchet. From secrecy to authenticity in security protocols. In *SAS '02: Proceedings of the 9th International Symposium on Static Analysis*, pages 342–359, London, UK, 2002. Springer-Verlag. (Cited on page 39.)

[74] Bruno Blanchet and Benjamin Aziz. A calculus for secure mobility. In *ASIAN*, pages 188–204, 2003. (Cited on page 39.)

[75] Valer Bocan. Threshold puzzles: The evolution of dos-resistant authenticated. Buletinul Stiintific al Universitatii âPolitehnicaâ din Timisoara, ROMANIA Seria AUTOMATICA si CALCULATOARE PERIODICA POLITECHNICA, Transactions on AUTOMATIC CONTROL and COMPUTER SCIENCE Vol.49 (63), 2004, ISSN 1224-600X. (Cited on page 91.)

[76] Valer Bocan and Mihai Fagadar-Cosma. Adaptive threshold puzzles. In *Computer as a Tool, 2005. EUROCON 2005.The International Conference on*, volume 1, pages 644 –647, nov. 2005. (Cited on page 91.)

[77] Yohan Boichut, Pierre-Cyrille Héam, and Olga Kouchnarenko. Automatic verification of security protocols using approximations. Research Report RR-5727, INRIA, 2005. (Cited on pages 32 and 57.)

[78] Michele Boreale. Symbolic trace analysis of cryptographic protocols. In *ICALP '01: Proceedings of the 28th International Colloquium on Automata, Languages and Programming,*, pages 667–681, London, UK, 2001. Springer-Verlag. (Cited on page 39.)

[79] Michele Boreale. Erratum of proof techniques for cryptographic processes., 2004. (unpublished manuscript). (Cited on page 38.)

[80] Johannes Borgström and Uwe Nestmann. On bisimulations for the spi calculus. *Mathematical. Structures in Comp. Sci.*, 15(3):487–552, 2005. (Cited on page 38.)

[81] Daniel Boteanu, José M. Fernandez, John McHugh, and John Mullins. Queue management as a dos counter-measure? In *Proceedings of the 10th international conference on Information Security*, ISC'07, pages 263–280, Berlin, Heidelberg, 2007. Springer-Verlag. (Cited on page 106.)

[82] Julien Bournelle, Maryline Laurent-Maknavicius, Hannes Tschofenig, and Yacine El Mghazli. Handover-aware access control mechanism: Ctp for pana. In Mário Freire, Prosper Chemouil, Pascal Lorenz, and Annie Gravey, editors, *Universal Multiservice Networks*, volume 3262 of *Lecture Notes in Computer Science*, pages 430–439. Springer Berlin / Heidelberg, 2004. 10.1007/978-3-540-30197-4_43. (Cited on page 54.)

[83] Colin Boyd and Anish Mathuria. *Protocols for Authentication and Key Establishment.* Springer, 2003. (Cited on page 23.)

[84] Stephen H. Brackin. A hol extension of gny for automatically analyzing cryptographic protocols. In *In Proceedings of the Ninth IEEE Computer Security Foundations Workshop*, pages 62–76. Press, 1996. (Cited on pages 6, 12 and 27.)

[85] Sébastien Briais and Uwe Nestmann. Open bisimulation, revisited. *Theor. Comput. Sci.*, 386(3):236–271, 2007. (Cited on page 38.)

[86] Hans Uttel Brics and Hans Hüttel. Deciding framed bisimilarity. In *In INFINITYâ02*, pages 1–20, 2002. (Cited on page 38.)

[87] Bob Briscoe, Arnaud Jacquet, Carla Di Cairano-Gilfedder, Alessandro Salvatori, Andrea Soppera, and Martin Koyabe. Policing congestion response in an internetwork using re-feedback. *SIGCOMM Comput. Commun. Rev.*, 35(4):277–288, August 2005. (Cited on page 86.)

[88] Aaron Brown and David A. Patterson. Towards availability benchmarks: a case study of software raid systems. In *Proceedings of the annual conference on USENIX Annual Technical Conference*, ATEC '00, pages 22–22, Berkeley, CA, USA, 2000. USENIX Association. (Cited on page 110.)

[89] Randal E. Bryant. Graph-based algorithms for boolean function manipulation. *IEEE Trans. Comput.*, 35:677–691, August 1986. (Cited on page 27.)

[90] Randal E. Bryant. Graph-based algorithms for boolean function manipulation. *IEEE Transactions on Computers*, 35:677–691, 1986. (Cited on page 33.)

[91] Randal E. Bryant. Symbolic boolean manipulation with ordered binary-decision diagrams. *ACM Comput. Surv.*, 24:293–318, September 1992. (Cited on page 27.)

[92] Michele Bugliesi and Giuseppe Castagna. Secure safe ambients. *SIGPLAN Not.*, 36(3):222–235, 2001. (Cited on page 39.)

[93] Michele Bugliesi, Giuseppe Castagna, and Silvia Crafa. Boxed ambients. In *In Proc. TACS 2001, LNCS 2215*, pages 38–63. Springer-Verlag, 2001. (Cited on page 39.)

[94] J. R. Burch, E. M. Clarke, K. L. Mcmillan, D. L. Dill, and L. J. Hwang. Symbolic model checking: 10^{20} states and beyond, 1990. (Cited on page 33.)

[95] Michael Burrows, Martin Abadi, and Roger Needham. A logic of authentication. *ACM Transactions on Computer Systems*, 8:18–36, 1990. (Cited on pages 6, 12, 23, 26 and 44.)

[96] L. Buttyan, S. Staamann, and U. Wilhelm. A simple logic for authentication protocol design. In *Computer Security Foundations Workshop, 1998. Proceedings. 11th IEEE*, pages 153–162, June 1998. (Cited on page 44.)

[97] Ran Canetti, Ling Cheung, Dilsun Kaynar, Moses Liskov, Nancy Lynch, Olivier Pereira, and Roberto Segala. Analyzing security protocols using time-bounded task-pioas. *Discrete Event Dynamic Systems*, 18(1):111–159, March 2008. (Cited on page 119.)

[98] Luca Cardelli and Andrew D. Gordon. Mobile ambients, 1998. (Cited on page 39.)

[99] Claude Castelluccia, Einar Mykletun, and Gene Tsudik. Improving secure server performance by re-balancing ssl/tls handshakes. In *ASIACCS '06: Proceedings of the 2006 ACM Symposium on Information, computer and communications security*, pages 26–34, New York, NY, USA, 2006. ACM. (Cited on pages 94, 95 and 96.)

[100] Jan Cederquist and Mohammad Torabi Dashti. An intruder model for verifying liveness in security protocols. In *FMSE '06: Proceedings of the fourth ACM workshop on Formal methods in security*, pages 23–32, New York, NY, USA, 2006. ACM. (Cited on page 20.)

[101] CERT. http://www.cert.org/tech_tips/denial_of_service.html. (Cited on page 19.)

[102] CERT. CERT Advisory CA-1996-21 TCP SYN Flooding and IP Spoofing Attacks. Available online at: http://www.cert.org/advisories/CA-1996-21.html. (Cited on page 80.)

[103] CERT. CERT Advisory Ca-1997-28: IP Denial-of-service Attacks. Available on: http://www.cert.org/advisories/CA-1997-28.html. (Cited on page 81.)

[104] CERT. Cert advisory ca-2001-20 continuing threats to home users. July 23, 2001, Available on http://www.cert.org/advisories/CA-2001-20.html. (Cited on page 83.)

[105] CERT. CERT Incident Note IN-2000-05: "mstream" Distributed Denial of Service Tool., 2000. Available on: http://www.cert.org/incident_notes/IN-2000-05.html. (Cited on page 83.)

[106] CERT. Cert/cc statistics (historical)., 2006. Available online on http://www.cert.org/stats/cert_stats.html#vul-total. (Cited on page 78.)

[107] Wu chang Feng. The case for tcp/ip puzzles. In *in ACM SIGCOMM Workshop on Future Directions in Network Architecture (FDNA03*, pages 322–327. ACM Press, 2003. (Cited on pages 88 and 94.)

[108] E.Y. Chen. Detecting dos attacks on sip systems. In *VoIP Management and Security, 2006. 1st IEEE Workshop on*, pages 53–58, 3 2006. (Cited on page 81.)

[109] Yan Chen, Adam W. Bargteil, David Bindel, Randy H. Katz, and John Kubiatowicz. Quantifying network denial of service: A location service case study. In *ICICS'01*, pages 340–351, 2001. (Cited on page 110.)

[110] Yan Chen, Toni Farley, and Nong Ye. Qos requirements of network applications on the internet. *Inf. Knowl. Syst. Manag.*, 4(1):55–76, January 2004. (Cited on page 111.)

[111] Yu Chen and Kai Hwang. Collaborative detection and filtering of shrew ddos attacks using spectral analysis. *J. Parallel Distrib. Comput.*, 66(9):1137–1151, September 2006. (Cited on page 84.)

[112] Y. Chevalier, L. Compagna, J. Cuellar, P. Hankes Drielsma, J. Mantovani, S. Mödersheim, and L. Vigneron. A high level protocol specification language for industrial security-sensitive protocols. In *Austrian Computer Society*, pages 193–205, 2004. (Cited on page 56.)

[113] Yannick Chevalier and Laurent Vigneron. Automated unbounded verification of security protocols. In *CAV '02: Proceedings of the 14th International Conference on Computer Aided Verification*, pages 324–337, London, UK, 2002. Springer-Verlag. (Cited on page 31.)

[114] Yannick Chevalier and Laurent Vigneron. Automated unbounded verification of security protocols. In *CAV '02: Proceedings of the 14th International Conference on Computer Aided Verification*, pages 324–337, London, UK, 2002. Springer-Verlag. (Cited on page 56.)

[115] Kim-Kwang Raymond Choo. An integrative framework to protocol analysis and repair: Bellare–rogaway model + planning + model checker, 2006. (Cited on page 46.)

[116] Wu chun Feng, A. Kapadia, and S. Thulasidasan. Green: proactive queue management over a best-effort network. In *Global Telecommunications Conference, 2002. GLOBECOM '02. IEEE*, volume 2, pages 1774 – 1778 vol.2, nov. 2002. (Cited on page 108.)

[117] Jae Chung and Mark Claypool. Dynamic-cbt and chips-router support for improved multimedia performance on the internet. In *Proceedings of the eighth ACM international conference on Multimedia*, MULTIMEDIA '00, pages 239–248, New York, NY, USA, 2000. ACM. (Cited on page 108.)

[118] A. Cimatti, E. Clarke, F. Giunchiglia, and M. Roveri. Nusmv: a new symbolic model checker. *International Journal on Software Tools for Technology Transfer*, 2:2000, 2000. (Cited on page 6.)

[119] Alessandro Cimatti, Edmund Clarke, Enrico Giunchiglia, Fausto Giunchiglia, Marco Pistore, Marco Roveri, Roberto Sebastiani, and Armando Tacchella. Nusmv 2: An opensource tool for symbolic model checking, 2002. (Cited on pages 6 and 33.)

[120] Alessandro Cimatti, Enrico Giunchiglia, Marco Pistore, Marco Roveri, Roberto Sebastiani, Armando Tacchella, and O Tacchella. Integrating bdd-based and sat-based symbolic model checking, 2002. (Cited on page 33.)

[121] CISCO. Defining Strategies to Protect Against TCP SYN Denial of Service Attacks. Available online at: http://www.cisco.com/en/US/tech/tk828/technologies_tech_note09186a00800f67d5.shtml. (Cited on page 80.)

[122] J.A. Clark and J.L. Jacob. Searching for a solution: engineering tradeoffs and the evolution of provably secure protocols. In *Security and Privacy, 2000. S P 2000. Proceedings. 2000 IEEE Symposium on*, 2000. (Cited on pages 12, 44 and 45.)

[123] John Clark. Attacking authentication protocols. *High Integrity Systems*, 1, 1996. (Cited on page 24.)

[124] John Clark and Jeremy Jacob. A survey of authentication protocol literature: Version 1.0. Technical report, 1997. (Cited on pages 23, 31, 46 and 72.)

[125] E. M. Clarke, O. Grumberg, and D. A. Peled. *Model Checking*. The MIT Press, 1999. (Cited on page 5.)

[126] E. M. Clarke, S. Jha, and W. Marrero. Verifying security protocols with brutus. *ACM Trans. Softw. Eng. Methodol.*, 9(4):443–487, 2000. (Cited on pages 29 and 30.)

[127] Edmund Clarke, Somesh Jha, and Will Marrero. Efficient verification of security protocols using partial-order reductions. *International Journal on Software Tools for Technology Transfer*, 4:173–188, 2003. (Cited on page 27.)

[128] Edmund M. Clarke, Somesh Jha, and Wilfredo R. Marrero. Partial order reductions for security protocol verification. In *Proceedings of the 6th International Conference on Tools and Algorithms for Construction and Analysis of Systems: Held as Part of the European Joint Conferences on the Theory and Practice of Software, ETAPS 2000*, TACAS '00, pages 503–518, London, UK, UK, 2000. Springer-Verlag. (Cited on page 27.)

[129] E.M. Clarke, E.A. Emerson, S. Jha, and A.P. Sistla. Symmetry reductions in model checking. In AlanJ. Hu and MosheY. Vardi, editors, *Computer Aided Verification*, volume 1427 of *Lecture Notes in Computer Science*, pages 147–158. Springer Berlin Heidelberg, 1998. (Cited on page 27.)

[130] Sebastian Clauss. A framework for quantification of linkability within a privacy-enhancing identity management system. In Günter Müller, editor, *Emerging Trends in Information and Communication Security*, volume 3995 of *Lecture Notes in Computer Science*, pages 191–205. Springer Berlin Heidelberg, 2006. 10.1007/11766155_14. (Cited on page 15.)

[131] Fabien Coelho. Exponential memory-bound functions for proof of work protocols. research report a-370, cri, Ãcole des mines de, 2005. Technical report 2005/356, http://eprint.iacr.org/2005/356. (Cited on page 118.)

[132] Ernie Cohen. Taps: A first-order verifier for cryptographic protocols. In *CSFW '00: Proceedings of the 13th IEEE workshop on Computer Security Foundations*, page 144, Washington, DC, USA, 2000. IEEE Computer Society. (Cited on page 40.)

[133] Ernie Cohen. First-order verification of cryptographic protocols. *J. Comput. Secur.*, 11(2):189–216, 2003. (Cited on page 40.)

[134] Hubert Comon and Robert Nieuwenhuis. Induction = i-axiomatization + first-order consistency. *Inf. Comput.*, 159(1-2):151–186, 2000. (Cited on page 41.)

[135] Fady Copty, Limor Fix, Ranan Fraer, Enrico Giunchiglia, Gila Kamhi, Armando Tacchella, and Moshe Y. Vardi. Benefits of bounded model checking at an industrial setting. In *CAV '01: Proceedings of the 13th International Conference on Computer Aided Verification*, pages 436–453, London, UK, 2001. Springer-Verlag. (Cited on page 33.)

[136] R. Corin, S. Etalle, P. H. Hartel, and A. Mader. Timed analysis of security protocols. *J. Comput. Secur.*, 15(6):619–645, December 2007. (Cited on page 119.)

[137] Ricardo Corin, Sandro Etalle, Pieter H. Hartel, and Angelika Mader. Timed analysis of security protocols. *CoRR*. (Cited on page 37.)

[138] Symantec Corporation. Symantec internet security threat report (trends for 2010), 2011. Volume 16, Published April 2011 https://www4.symantec.com/mktginfo/downloads/21182883_GA_REPORT_ISTR_Main-Report_04-11_HI-RES.pdf. (Cited on page 78.)

[139] Véronique Cortier, Stéphanie Delaune, and Pascal Lafourcade. A survey of algebraic properties used in cryptographic protocols. *J. Comput. Secur.*, 14:1–43, January 2006. (Cited on page 49.)

[140] Olivier Coudert, Christian Berthet, and Jean Madre. Verification of synchronous sequential machines based on symbolic execution. In Joseph Sifakis, editor, *Automatic Verification Methods for Finite State Systems*, volume 407 of *Lecture Notes in Computer Science*, pages 365–373. Springer Berlin / Heidelberg, 1990. 10.1007/3-540-52148-8_30. (Cited on page 27.)

[141] C. J. F. Cremers, S. Mauw, and E. P. de Vink. Injective synchronisation: an extension of the authentication hierarchy. *Theor. Comput. Sci.*, 367:139–161, November 2006. (Cited on page 49.)

[142] Cas Cremers and Sjouke Mauw. Operational semantics of security protocols. In *Scenarios: Models, Transformations and Tools, International Workshop, Dagstuhl*, pages 66–89. Springer, 2005. (Cited on page 36.)

[143] Cas J. Cremers. The scyther tool: Verification, falsification, and analysis of security protocols. In *Proceedings of the 20th international conference on Computer Aided Verification*, CAV '08, pages 414–418, Berlin, Heidelberg, 2008. Springer-Verlag. (Cited on pages 6 and 36.)

[144] Cas J. Cremers, Pascal Lafourcade, and Philippe Nadeau. Formal to practical security. chapter Comparing State Spaces in Automatic Security Protocol Analysis, pages 70–94. Springer-Verlag, Berlin, Heidelberg, 2009. (Cited on page 50.)

[145] Cas J.F. Cremers. Unbounded verification, falsification, and characterization of security protocols by pattern refinement. In *CCS '08: Proceedings of the 15th ACM conference on Computer and communications security*, pages 119–128, New York, NY, USA, 2008. ACM. (Cited on page 36.)

[146] C.J.F. Cremers. *Scyther - Semantics and Verification of Security Protocols*. Ph.D. dissertation, Eindhoven University of Technology, 2006. (Cited on page 36.)

[147] Wei Dai. Crypto++ 5.6.0 benchmarks. (Cited on page 118.)

[148] Nitish Dalal, Jenny Shah, Khushboo Hisaria, and Devesh Jinwala. A comparative analysis of tools for verification of security protocols. *IJCNS*, 3(10):779–787, 2010. (Cited on page 50.)

[149] Anupam Datta, Ante Derek, John C. Mitchell, and Dusko Pavlovic. A derivation system for security protocols and its logical formalization. In *In Proceedings of 16th IEEE Computer Security Foundations Workshop*, pages 109–125. IEEE, 2003. (Cited on page 129.)

[150] Stépphanie Delaune, Hai Lin, and Christopher Lynch. Protocol verification via rigid/flexible resolution. In *Logic for Programming, Artificial Intelligence, and Reasoning*, volume Volume 4790/2007, pages 242–256. Springer Berlin / Heidelberg, 2007. (Cited on page 41.)

[151] Xidong Deng, Sungwon Yi, G. Kesidis, and C.R. Das. Stabilized virtual buffer (svb) - an active queue management scheme for internet quality-of-service. In *Global Telecommunications Conference, 2002. GLOBECOM '02. IEEE*, volume 2, pages 1628 – 1632 vol.2, nov. 2002. (Cited on page 108.)

[152] Sven Dietrich, Neil Long, and David Dittrich. An analysis of the "shaft" distributed denial of service tool., 2000. Available on: http://home.adelphi.edu/špock/shaft
_analysis.txt. (Cited on page 83.)

[153] Sven Dietrich, Neil Long, and David Dittrich. Analyzing distributed denial of service tools: The shaft case. In *Proceedings of the 14th USENIX conference on System administration*, LISA '00, pages 329–340, Berkeley, CA, USA, 2000. USENIX Association. (Cited on page 109.)

[154] David Dittrich. The DoS Project's "trinoo" distributed denial of service attack tool., 1999. http://staff.washington.edu/dittrich/misc/ddos/. (Cited on page 83.)

[155] David Dittrich. The "stacheldraht" distributed denial of service attack tool., 1999. Available on : http://staff.washington.edu/dittrich/misc/stacheldraht.analysis. (Cited on page 83.)

[156] David Dittrich. The "tribe flood network" distributed denial of service attack tool., 1999. http://staff.washington.edu/dittrich/misc/tfn.analysis.txt. (Cited on page 83.)

[157] David Dittrich, George Weaver, Sven Dietrich, and Neil Long. The "mstream" distributed denial of service attack tool., 2000. Available on: http://staff.washington.edu/dittrich/misc/mstream.analysis.txt. (Cited on page 83.)

[158] Shaddin F. Doghmi, Joshua D. Guttman, and F. Javier Thayer. Authentication tests. In *In Proceedings, 2000 IEEE Symposium on Security and Privacy. May, IEEE Computer*, pages 96–109. Society Press, 2000. (Cited on page 35.)

[159] Shaddin F. Doghmi, Joshua D. Guttman, and F. Javier Thayer. Skeletons and the shapes of bundles. Technical report, MTR 05 B 02, The MITRE Corp, 2005. (Cited on page 36.)

[160] Shaddin F. Doghmi, Joshua D. Guttman, and F. Javier Thayer. Completeness of the authentication tests. In *ESORICS*, pages 106–121, 2007. (Cited on page 36.)

[161] Shaddin F. Doghmi, Joshua D. Guttman, and F. Javier Thayer. Searching for shapes in cryptographic protocols. In *Proceedings of the 13th international conference on Tools and algorithms for the construction and analysis of systems*, TACAS'07, pages 523–537, Berlin, Heidelberg, 2007. Springer-Verlag. (Cited on page 36.)

[162] Danny Dolev and Andrew C. Yao. On the security of public key protocols. Technical report, Stanford, CA, USA, 1981. (Cited on pages 6, 20, 21, 25, 27, 96 and 118.)

[163] Cynthia Dwork, Andrew Goldberg, and Moni Naor. On memory-bound functions for fighting spam. In *In Crypto*, pages 426–444. Springer-Verlag, 2002. (Cited on page 86.)

[164] Cynthia Dwork and Moni Naor. Pricing via processing or combatting junk mail. In *Proceedings of the 12th Annual International Cryptology Conference on Advances in Cryptology*, CRYPTO '92, pages 139–147, London, UK, UK, 1993. Springer-Verlag. (Cited on pages 86, 87 and 95.)

[165] Anders Strandlϕv Elkjaer, Michael Hoehle, , Hans Huettel, and Kasper Overgard Nielsen. Towards automatic bisimilarity checking in the spi calculus, 1999. (Cited on page 38.)

[166] E. Allen Emerson and Thomas Wahl. On combining symmetry reduction and symbolic representation for efficient model checking. In *In Conference on Correct Hardware Design and Verification Methods (CHARME*, pages 216–230. Springer, 2003. (Cited on page 27.)

[167] F. Javier Thayer Fabrega, Jonathan C. Herzog, and Joshua D. Guttman. Strand spaces: Why is a security protocol correct?, 1998. (Cited on page 34.)

[168] Mehran Fallah. A puzzle-based defense strategy against flooding attacks using game theory. *IEEE Trans. Dependable Secur. Comput.*, 7:5–19, January 2010. (Cited on page 92.)

[169] Wu-chang Feng, Kang G. Shin, Dilip D. Kandlur, and Debanjan Saha. The blue active queue management algorithms. *IEEE/ACM Trans. Netw.*, 10(4):513–528, August 2002. (Cited on page 107.)

[170] P. Ferguson and D. Senie. Network ingress filtering: Defeating denial of service attacks which employ ip source address spoofing, 2000. (Cited on page 83.)

[171] Richard H. Fifarek. Linux.slapper.worm: Buffer overflow attacks continue to be a problem. SANS GSEC Practical ver. 1.4b, Option 1, Oct. 1, 2002. (Cited on page 19.)

[172] Cormac Flanagan and Shaz Qadeer. Assume-guarantee model checking. Technical report, 2003. (Cited on page 47.)

[173] Sally Floyd and Van Jacobson. Random early detection gateways for congestion avoidance. *IEEE/ACM Trans. Netw.*, 1:397–413, August 1993. (Cited on pages 106 and 107.)

[174] Riccardo Focardi, Roberto Gorrieri, Ruggero Lanotte, Andrea Maggiolo-Schettini, Fabio Martinelli, Simone Tini, and Enrico Tronci. (Cited on page 119.)

[175] Simon N. Foley. Towards a framework for autonomic security protocols. In Bruce Christianson, Bruno Crispo, James A. Malcolm, and Michael Roe, editors, *Security Protocols*, volume 3364 of *Lecture Notes in Computer Science*, pages 55–62. Springer Berlin / Heidelberg, 2005. 10.1007/11542322_9. (Cited on page 44.)

[176] Y. Forsberg, D. andOhba, B. Patil, H. Tschofenig, and A. Yegin. Protocol for carrying authentication for network access (pana), 2008. RFC 5191 URL: http://tools.ietf.org/html/rfc5191, May 2008. (Cited on pages 53 and 54.)

[177] M. Franklin and M. Reiter. A linear protocol failure for rsa with exponent three. In *CRYPTO '95 Rump Session*, 1995. (Cited on page 19.)

[178] Yi Gao. Efficient trapdoor-based client puzzle system against dos attacks. Master of Computer Science by Research, School of InformationTechnology and Computer Science, University of Wollongong, Wollongong, Australia, 2005. (Cited on pages 86 and 89.)

[179] Yi Gao, Willy Susilo, Yi Mu, and Jennifer Seberry. Efficient trapdoor-based client puzzle against dos attacks. In Scott C.-H. C.-H. Huang, David MacCallum, and Ding-Zhu Du, editors, *Network Security*, pages 229–249. Springer US, 2010. 10.1007/978-0-387-73821-5_10. (Cited on page 89.)

[180] Lee Garber. Denial-of-service attacks rip the internet. *Computer*, 33(4):12–17, 2000. (Cited on page 77.)

[181] Thomas Genet. Decidable approximations of sets of descendants and sets of normal forms. In *In Proc. 9th RTA Conf., Tsukuba (Japan), volume 1379 of LNCS*, pages 151–165. Springer-Verlag, 1997. (Cited on page 32.)

[182] Thomas Genet and Francis Klay. Rewriting for cryptographic protocol verification. In *CADE-17: Proceedings of the 17th International Conference on Automated Deduction*, pages 271–290, London, UK, 2000. Springer-Verlag. (Cited on page 32.)

[183] Bela Genge, Iosif Ignat, and Piroska Haller. Automated composition of security protocols. *CoRR*, abs/0908.4325, 2009. informal publication. (Cited on page 45.)

[184] Thomer M. Gil and Massimiliano Poletto. Multops: a data-structure for bandwidth attack detection. In *SSYM'01: Proceedings of the 10th conference on USENIX Security Symposium*, pages 3–3, Berkeley, CA, USA, 2001. USENIX Association. (Cited on page 85.)

[185] Virgil D. Gligor. A note on denial-of-service in operating systems. *IEEE Trans. Softw. Eng.*, 10(3):320–324, 1984. (Cited on page 76.)

[186] Yann Glouche, Thomas Genet, Olivier Heen, and Olivier Courtay. A security protocol animator tool for AVISPA. In *In ARTIST-2 workshop*, 2006. (Cited on page 57.)

[187] Li Gong, Roger Needham, and Raphael Yahalom. Reasoning about belief in cryptographic protocols. In *Proceedings 1990 IEEE Symposium on Research in Security and Privacy*, pages 234–248. IEEE Computer Society Press, 1990. (Cited on page 27.)

[188] Li Gong and Paul Syverson. Fail-stop protocols: An approach to designing secure protocols. In *Dependable Computing for Critical Applications 5*, pages 44–55. IEEE Computer Society, 1995. (Cited on pages 43 and 99.)

[189] Andrew D. Gordon and Alan Jeffrey. Authenticity by typing for security protocols. In *Journal of Computer Security*, pages 145–159. IEEE Computer Society, 2001. (Cited on page 39.)

[190] Andrew D. Gordon and Alan Jeffrey. Types and effects for asymmetric cryptographic protocols. *J. Comput. Secur.*, 12(3,4):435–483, 2004. (Cited on page 39.)

[191] Jean Goubault-Larrecq, Catuscia Palamidessi, and Angelo Troina. A probabilistic applied pi-calculus. In *Porc. of APLAS'07, 5th Asian Symposium on Programming Languages and Systems*, volume 4870 of *LNCS*, pages 175–190. Springer, 2007. (Cited on page 39.)

[192] Bogdan Groza and Dorina Petrica. On chained cryptographic puzzles. *Budapesti Műszaki Főiskola*, 2006. (Cited on page 90.)

[193] J.D. Guttman and F.J. Thayer. Authentication tests. In *Security and Privacy, 2000. S P 2000. Proceedings. 2000 IEEE Symposium on*, 2000. (Cited on page 44.)

[194] Joshua D. Guttman. Security protocol design via authentication tests. In *In Proceedings of 15th IEEE Computer Security Foundations Workshop. IEEE Computer*, pages 92–103. Society Press, 2002. (Cited on pages 35 and 44.)

[195] ITU-T Recommendation H.530:. Symmetric security procedures for h.510 (mobility for h.323 multimedia systems and services) (2002)., 2002. (Cited on page 31.)

[196] Christian Haack and Alan Jeffrey. Timed spi-calculus with types for secrecy and authenticity. pages 202–216, 2005. (Cited on page 39.)

[197] Christian Haack and Alan Jeffrey. Pattern-matching spi-calculus. *Inf. Comput.*, 204(8):1195–1263, 2006. (Cited on page 39.)

[198] Chen Hao. A search-based framework for security protocol synthesis. (YCST-2007-12), 2007. (Cited on page 45.)

[199] D. Harkins and D. Carrel. The internet key exchange (ike). Standards Track RFC 2409, IETF, Nov 1998. http://www.ietf.org/rfc/rfc2409.txt. (Cited on page 94.)

[200] Eman Salaheddin Hashem. Analysis of random drop for gateway congestion control. report LCS, TR-465,Laboratory for Computer Science, MIT ,Cambridge, MA,1989. (Cited on page 107.)

[201] James Heather, Gavin Lowe, and Steve Schneider. How to prevent type flaw attacks on security protocols. *J. Comput. Secur.*, 11:217–244, March 2003. (Cited on page 19.)

[202] C.V. Hollot, V. Misra, D. Towsley, and Wei-Bo Gong. On designing improved controllers for aqm routers supporting tcp flows. In *INFOCOM 2001. Twentieth Annual Joint Conference of the IEEE Computer and Communications Societies. Proceedings. IEEE*, volume 3, pages 1726 –1734 vol.3, 2001. (Cited on page 108.)

[203] Gerard Holzmann. *Spin model checker, the: primer and reference manual*. Addison-Wesley Professional, first edition, 2003. (Cited on page 6.)

[204] J. Howard., 1998. An Analysis of Security Incidents on the Internet. PhD thesis, Carnegie Mellon University, Aug. 1998. (Cited on page 77.)

[205] Antti Huima. Efficient infinite-state analysis of security protocols. In *Proc. FLOC'99 Workshop on Formal Methods and Security Protocols*, 1999. (Cited on page 39.)

[206] Alefiya Hussain, John Heidemann, and Christos Papadopoulos. A framework for classifying denial of service attacks. In *SIGCOMM '03: Proceedings of the 2003 conference on Applications, technologies, architectures, and protocols for computer communications*, pages 99–110, New York, NY, USA, 2003. ACM. (Cited on pages 76 and 79.)

[207] Alefiya Hussain, John Heidemann, and Christos Papadopoulos. A framework for classifying denial of service attacks. In *Proceedings of the 2003 conference on Applications, technologies, architectures, and protocols for computer communications*, SIGCOMM '03, pages 99–110, New York, NY, USA, 2003. ACM. (Cited on page 84.)

[208] Alefiya Hussain, John Heidemann, and Christos Papadopoulos. Identification of repeated denial of service attacks. In *Proceedings of the IEEE Infocom*, page to appear, Barcelona, Spain, April 2006. IEEE. (Cited on page 84.)

[209] M. Mitchell J. Mitchell and U. Stern. Automated analysis of cryptographic protocols using murϕ. *IEEE Symposium on Security and Privacy.*, 1997. (Cited on page 29.)

[210] Florent Jacquemard, Michaël Rusinowitch, and Laurent Vigneron. Compiling and verifying security protocols. 2000. (Cited on page 31.)

[211] Gizela Jakubowska and Wojciech Penczek. Modelling and checking timed authentication of security protocols. *Fundam. Inf.*, 79(3-4):363–378, August 2007. (Cited on page 119.)

[212] Kurt Jensen, Lars Michael, Kristensen Lisa Wells, K. Jensen, and L. M. Kristensen. Coloured petri nets and cpn tools for modelling and validation of concurrent systems. In *International Journal on Software Tools for Technology Transfer*, 2007. (Cited on page 6.)

[213] Wang Juan, Zhou Yajie, and Zhang Huanguo. The simplified inductive approach to verifying security protocols. In *ISECS '08: Proceedings of the 2008 International Symposium on Electronic Commerce and Security*, pages 523–526, Washington, DC, USA, 2008. IEEE Computer Society. (Cited on page 43.)

[214] Juan C. Lopez P, Raúl Monroy, and Dieter Hutter. On the automated correction of security protocols susceptible to a replay attack. In *Computer Security – ESORICS 2007*, volume Volume 4734/2008, pages 594–609. Springer Berlin / Heidelberg, 2007. (Cited on pages 12 and 45.)

[215] Judson Santos Santiago and Laurent Vigneron. Optimistic non-repudiation protocol analysis. In *Information Security Theory and Practices - Smart Cards, Mobile and Ubiquitous Computing Systems First IFIP TC6 / WG 8.8 / WG 11.2 International Workshop, WISTP 2007, Heraklion, Crete, Greece,*

May 9-11, 2007. Proceedings Lecture Notes in Computer Science, volume 4462 of *Lecture Notes in Computer Science*, pages 90–101. Springer, 2007. (Cited on page 23.)

[216] Ari Juels and John G. Brainard. Client puzzles: A cryptographic countermeasure against connection depletion attacks. In *NDSS*, 1999. (Cited on pages 83, 86, 87, 88, 94, 95 and 106.)

[217] P. Karn and W. A. Simpson. Photuris: Session-key management protocol. Experimental RFC 2522, IETF, Mar 1999. http://www.ietf.org/rfc/rfc2522.txt. (Cited on pages 94, 95 and 96.)

[218] C. Kaufman. Internet key exchange (ikev2) protocol. (Cited on page 96.)

[219] Moore J.S.: Kaufmann, M. Some key research problems in automated theorem proving for hardware and software verification. (Cited on page 47.)

[220] Frank Scalzo Ken Silva and Piet Barber. Anatomy of recent dns reflector attacks from the victim and reflector point of view., April 2006. White paper. available on http://www.verisign.com/static/037903.pdf. (Cited on pages 77 and 82.)

[221] Kapila S. Khurana V Yadav V Shorey R Saran H Juneja S. Kamra A. Sfed: a rate control based active queue management discipline. IBM India Research Laboratory Research Report, 2000. (Cited on page 107.)

[222] Il-Gon Kim and Jin-Young Choi. Formal verification of PAP and EAP-MD5 protocols in wireless networks: FDR model checking. In *Proceedings of the 18th International Conference on Advanced Information Networking and Applications - Volume 2*, AINA '04, pages 264–, Washington, DC, USA, 2004. IEEE Computer Society. (Cited on page 72.)

[223] Steve Kremer and Laurent Mazare. Adaptive soundness of static equivalence. In *In Proceedings of the 12th European Symposium on Research in Computer Security, ESORICS'07, edited by Biskup, J. and Lopez, J., LNCS Vol. 4734*, pages 610–625, 2007. (Cited on page 22.)

[224] Amit Kulkarni and Stephen Bush. Detecting distributed denial-of-service attacks using kolmogorov complexity metrics. *J. Netw. Syst. Manage.*, 14(1):69–80, March 2006. (Cited on page 84.)

[225] Srisankar S. Kunniyur and R. Srikant. An adaptive virtual queue (avq) algorithm for active queue management. *IEEE/ACM Trans. Netw.*, 12(2):286–299, apr 2004. (Cited on page 108.)

[226] Ralf Küsters and Tomasz Truderung. Using proverif to analyze protocols with diffie-hellman exponentiation. In *Proceedings of the 2009 22nd IEEE Computer Security Foundations Symposium*, pages 157–171, Washington, DC, USA, 2009. IEEE Computer Society. (Cited on page 49.)

[227] Ralf Küsters and Tomasz Truderung. Reducing protocol analysis with xor to the xor-free case in the horn theory based approach. *Journal of Automated Reasoning*, 46:325–352, 2011. 10.1007/s10817-010-9188-8. (Cited on page 49.)

[228] Aleksandar Kuzmanovic and Edward W. Knightly. Low-rate tcp-targeted denial of service attacks: the shrew vs. the mice and elephants. In *Proceedings of the 2003 conference on Applications, technologies, architectures, and protocols for computer communications*, SIGCOMM '03, pages 75–86, New York, NY, USA, 2003. ACM. (Cited on page 119.)

[229] Pascal Lafourcade, Vanessa Terrade, and Sylvain Vigier. Comparison of cryptographic verification tools dealing with algebraic properties. In *Formal Aspects in Security and Trust*, pages 173–185, 2009. (Cited on page 50.)

[230] Stéphane Lafrance and John Mullins. An information flow method to detect denial of service vulnerabilities. In *Formal Specifications of Computer-Based Systems, volume 9, pages 1258– 1260. Journal of Universal Computer Science*, 2003. (Cited on pages 97, 99, 100 and 101.)

[231] David Lapsley and Steven Low. Random early marking: An optimization approach to internet congestion control. In *Proceedings of the 7th IEEE International Conference on Networks*, ICON '99, pages 67–, Washington, DC, USA, 1999. IEEE Computer Society. (Cited on page 108.)

[232] Martin P. Loeb William Lucyshyn Lawrence A. Gordon and Robert Richardson. 2005 CSI/FBI Computer Crime and Security Survey., 2005. (Cited on page 78.)

[233] M. C. Lee and Chun-Kan Fung. A public-key based authentication and key establishment protocol coupled with a client puzzle. *J. Am. Soc. Inf. Sci. Technol.*, 54(9):810–823, 2003. (Cited on page 96.)

[234] Jussipekka Leiwo, Tuomas Aura, and Pekka Nikander. Towards network denial of service resistant protocols. In *Proceedings of the IFIP TC11 Fifteenth Annual Working Conference on Information Security for Global Information Infrastructures*, pages 301–310, Deventer, The Netherlands, The Netherlands, 2000. Kluwer, B.V. (Cited on page 94.)

[235] Jonathan Lemon. Resisting syn flood dos attacks with a syn cache. In *BSDC'02: Proceedings of the BSD Conference 2002 on BSD Conference*, pages 10–10, Berkeley, CA, USA, 2002. USENIX Association. (Cited on pages 94 and 95.)

[236] A. K. Lenstra, H. W. Lenstra, and L. Lovász. Factoring polynomials with rational coefficients. *Mathematische Annalen*, 261:515–534, 1982. 10.1007/BF01457454. (Cited on page 91.)

[237] Francesca Levi and Davide Sangiorgi. Controlling interference in ambients. In *POPL '00: Proceedings of the 27th ACM SIGPLAN-SIGACT symposium on Principles of programming languages*, pages 352–364, New York, NY, USA, 2000. ACM. (Cited on page 39.)

[238] John Leyden. UK teenager accused of electronic sabotage against US port., 2003. Available on http://www.theregister.co.uk/2003/10/06/uk_teenager_accused_of_electronic/. (Cited on page 77.)

[239] Jun Li, Jelena Mirkovic, Mengqiu Wang, Peter L. Reiher, and Lixia Zhang. Save: Source address validity enforcement protocol. In *INFOCOM*, 2002. (Cited on page 83.)

[240] Chu-Hsing Lin, Jung-Chun Liu, Fuu-Cheng Jiang, and Chien-Ting Kuo. An effective priority queue-based scheme to alleviate malicious packet flows from distributed dos attacks. In *Proceedings of the 2008 International Conference on Intelligent Information Hiding and Multimedia Signal Processing*, IIH-MSP '08, pages 1371–1374, Washington, DC, USA, 2008. IEEE Computer Society. (Cited on page 108.)

[241] Dong Lin and Robert Morris. Dynamics of random early detection. *SIGCOMM Comput. Commun. Rev.*, 27(4):127–137, October 1997. (Cited on page 107.)

[242] Xin Liu, Ang Li, Xiaowei Yang, and David Wetherall. Passport: secure and adoptable source authentication. In *Proceedings of the 5th USENIX Symposium on Networked Systems Design and Implementation*, NSDI'08, pages 365–378, Berkeley, CA, USA, 2008. USENIX Association. (Cited on page 83.)

[243] Xin Liu, Xiaowei Yang, and Yanbin Lu. To filter or to authorize: network-layer dos defense against multimillion-node botnets. In *Proceedings of the ACM SIGCOMM 2008 conference on Data communication*, SIGCOMM '08, pages 195–206, New York, NY, USA, 2008. ACM. (Cited on page 85.)

[244] Xin Liu, Xiaowei Yang, and Yong Xia. Netfence: preventing internet denial of service from inside out. *SIGCOMM Comput. Commun. Rev.*, 41(4):–, August 2010. (Cited on page 86.)

[245] Chengnian Long, Bin Zhao, Xinping Guan, and Jun Yang. The yellow active queue management algorithm. *Comput. Netw. ISDN Syst.*, 47(4):525–550, March 2005. (Cited on page 107.)

[246] J. Loughney, M. Nakhjiri, C. Perkins, and R. Koodli. Context transfer protocol (cxtp), 2005. RFC 4067, URL: http://tools.ietf.org/html/rfc4067. (Cited on pages 54 and 55.)

[247] G. Lowe. A hierarchy of authentication specifications. In *Computer Security Foundations Workshop, 1997. Proceedings., 10th*, pages 31 –43, jun 1997. (Cited on page 49.)

[248] Gavin Lowe. An attack on the needham-schroeder public-key authentication protocol. *Information Processing Letters*, 56:131–133, 1995. (Cited on pages 22 and 26.)

[249] Gavin Lowe. Breaking and fixing the needham-schroeder public-key protocol using fdr. In *TACAs '96: Proceedings of the Second International Workshop on Tools and Algorithms for Construction and Analysis of Systems*, pages 147–166, London, UK, 1996. Springer-Verlag. (Cited on pages 12 and 28.)

[250] Gavin Lowe. Casper: A compiler for the analysis of security protocols. In *Journal of Computer Security*, pages 53–84. Society Press, 1998. (Cited on pages 6 and 28.)

[251] Miao Ma. Mitigating denial of service attacks with password puzzles. In *ITCC '05: Proceedings of the International Conference on Information Technology: Coding and Computing (ITCC'05) - Volume II*, pages 621–626, Washington, DC, USA, 2005. IEEE Computer Society. (Cited on page 90.)

[252] Bharat B. Madan, Katerina Goseva-Popstojanova, Kalyanaraman Vaidyanathan, and Kishor S. Trivedi. Modeling and quantification of security attributes of software systems. In *DSN '02: Proceedings of the 2002 International Conference on Dependable Systems and Networks*, pages 505–514, Washington, DC, USA, 2002. IEEE Computer Society. (Cited on pages 98, 101 and 118.)

[253] Ratul Mahajan, Steven M. Bellovin, Sally Floyd, John Ioannidis, Vern Paxson, and Scott Shenker. Controlling high bandwidth aggregates in the network. *SIGCOMM Comput. Commun. Rev.*, 32(3):62–73, July 2002. (Cited on page 85.)

[254] Ajay Mahimkar and Vitaly Shmatikov. Game-based analysis of denial-of-service prevention protocols. In *CSFW '05: Proceedings of the 18th IEEE workshop on Computer Security Foundations*, pages 287–301, Washington, DC, USA, 2005. IEEE Computer Society. (Cited on pages 97 and 101.)

[255] A. M. Mathuria, R. Safavi-naini, and P. R. Nickolas. On the automation of gny logic. In *In Proceedings of the 18th Australian Computer Science Conference*, pages 370–379, 1995. (Cited on page 27.)

[256] K. Matsuura and H. Imai. Modification of internet key exchange resistant against denial-of-service. In Pre-Proceeding of Internet Workshop 2000 (IWS2000), pages 167–174, Feb 2000. (Cited on pages 94 and 96.)

[257] K.L. McMillan. Symbolic model checking, 1993. Kluwer Academic Publ. (Cited on page 33.)

[258] Catherine Meadows. Analyzing the needham-schroeder public-key protocol: A comparison of two approaches. In *Proceedings of the 4th European Symposium on Research in Computer Security: Computer Security*, ESORICS '96, pages 351–364, London, UK, 1996. Springer-Verlag. (Cited on page 50.)

[259] Catherine Meadows. The nrl protocol analyzer: An overview. *Journal of Logic Programming*, 26:113–131, 1996. (Cited on pages 6, 12 and 20.)

[260] Catherine Meadows. A formal framework and evaluation method for network denial of service. In *CSFW '99: Proceedings of the 12th IEEE workshop on Computer Security Foundations*, page 4, Washington, DC, USA, 1999. IEEE Computer Society. (Cited on pages 94, 95, 97 and 118.)

[261] Catherine Meadows. A cost-based framework for analysis of denial of service in networks. *J. Comput. Secur.*, 9(1-2):143–164, 2001. (Cited on pages 3, 10, 94, 95, 97, 99, 100, 101, 118, 123, 127 and 132.)

[262] Michael Menth, Rüdiger Martin, and Joachim Charzinski. Capacity overprovisioning for networks with resilience requirements. In *Proceedings of the 2006 conference on Applications, technologies, architectures, and protocols for computer communications*, SIGCOMM '06, pages 87–98, New York, NY, USA, 2006. ACM. (Cited on page 86.)

[263] J.K. Millen, S.C. Clark, and S.B. Freedman. The interrogator: Protocol secuity analysis. *Software Engineering, IEEE Transactions on*, SE-13(2):274 – 288, feb. 1987. (Cited on page 20.)

[264] R. Milner. *A Calculus of Communicating Systems*. Springer-Verlag New York, Inc., Secaucus, NJ, USA, 1982. (Cited on page 6.)

[265] Robin Milner. *Communicating and mobile systems - the Pi-calculus*. Cambridge University Press, 1999. (Cited on page 6.)

[266] Robin Milner, Joachim Parrow, and David Walker. A calculus of mobile processes, parts *I* and *II*. *Information and Computation* ., pages 1–40 and 41–77, September 1992. (Cited on page 37.)

[267] Robin Milner and Davide Sangiorgi. Barbed bisimulation. In *ICALP '92: Proceedings of the 19th International Colloquium on Automata, Languages and Programming*, pages 685–695, London, UK, 1992. Springer-Verlag. (Cited on page 38.)

[268] Jelena Mirkovic, Sven Dietrich, David Dittrich, and Peter. Reiher. Internet denial of service: Attack and defense mechanisms., 2005. (Cited on pages 76, 82 and 106.)

[269] Jelena Mirkovic, Gregory Prier, and Peter L. Reiher. Attacking ddos at the source. In *Proceedings of the 10th IEEE International Conference on Network Protocols*, ICNP '02, pages 312–321, Washington, DC, USA, 2002. IEEE Computer Society. (Cited on page 85.)

[270] Jelena Mirkovic and Peter Reiher. A taxonomy of ddos attack and ddos defense mechanisms. *SIGCOMM Comput. Commun. Rev.*, 34(2):39–53, 2004. (Cited on page 106.)

[271] Jelena Mirkovic and Peter Reiher. D-ward: A source-end defense against flooding denial-of-service attacks. *IEEE Trans. Dependable Secur. Comput.*, 2(3):216–232, July 2005. (Cited on page 85.)

[272] John C. Mitchell, Vitaly Shmatikov, and Ulrich Stern. Finite-state analysis of ssl 3.0. In *In Seventh USENIX Security Symposium*, pages 201–216, 1998. (Cited on page 29.)

[273] Matthew W. Moskewicz and Conor F. Madigan. Chaff: Engineering an efficient sat solver, 2001. (Cited on page 33.)

[274] R. Moskowitz. The Host Identity Protocol (HIP). Internet Draft,Internet Engineering Task Force, Feb 2007. Available on http://www.ietf.org/internet-drafts/draft-ietf-hip-base-07.txt. (Cited on pages 87 and 96.)

[275] Harikrishna Narasimhan, Venkatanathan Varadarajan, and C. Rangan. Game theoretic resistance to denial of service attacks using hidden difficulty puzzles. In Jin Kwak, Robert Deng, Yoojae Won, and Guilin Wang, editors, *Information Security, Practice and Experience*, volume 6047 of *Lecture Notes in Computer Science*, pages 359–376. Springer Berlin / Heidelberg, 2010. 10.1007/978-3-642-12827-1_26. (Cited on page 92.)

[276] Roger M. Needham. Denial of service: an example. *Commun. ACM*, 37(11):42–46, 1994. (Cited on pages 76 and 79.)

[277] Roger M. Needham and Michael D. Schroeder. Using encryption for authentication in large networks of computers. *Commun. ACM*, 21(12):993–999, 1978. (Cited on pages 25 and 72.)

[278] David M. Nicol, William H. Sanders, and Kishor S. Trivedi. Model-based evaluation: From dependability to security. *IEEE Trans. Dependable Secur. Comput.*, 1(1):48–65, 2004. (Cited on page 96.)

[279] Peter Ochsenschläger, Jürgen Repp, Roland Rieke, and Ulrich Nitsche. The sh-verification tool - abstraction-based verification of co-operating systems. *Formal Asp. Comput.*, 10(4):381–404, 1998. (Cited on page 46.)

[280] Hitoshi Ohsaki and Toshinori Takai. Actas : A system design for associative and commutative tree automata theory. *Electronic Notes in Theoretical Computer Science*, 124(1):97–111, March 2005. (Cited on page 32.)

[281] T.J. Ott, T.V. Lakshman, and L.H. Wong. Sred: stabilized red. In *INFOCOM '99. Eighteenth Annual Joint Conference of the IEEE Computer and Communications Societies. Proceedings. IEEE*, volume 3, pages 1346 –1355 vol.3, mar 1999. (Cited on page 108.)

[282] Rong Pan, B. Prabhakar, and K. Psounis. Choke - a stateless active queue management scheme for approximating fair bandwidth allocation. In *INFOCOM 2000. Nineteenth Annual Joint Conference of the IEEE Computer and Communications Societies. Proceedings. IEEE*, volume 2, pages 942 –951 vol.2, 2000. (Cited on page 107.)

[283] Kihong Park and Heejo Lee. On the effectiveness of route-based packet filtering for distributed dos attack prevention in power-law internets. In *In Proc. ACM SIGCOMM*, pages 15–26, 2001. (Cited on page 83.)

[284] Mark Parris, Kevin Jeffay, and F. Donelson Smith. Lightweight active router-queue management for multimedia networking. In *Multimedia Computing and Networking, SPIE Proceedings Series*, pages 162–174, 1999. (Cited on page 108.)

[285] Lawrence C. Paulson. Relations between secrets: Two formal analyses of the yahalom protocol. *Journal of Computer Security*, 9:2001, 1997. (Cited on page 42.)

[286] Lawrence C. Paulson. The inductive approach to verifying cryptographic protocols. *J. Comput. Secur.*, 6(1-2):85–128, 1998. (Cited on pages 6, 12, 30, 40 and 42.)

[287] Lawrence C. Paulson. Inductive analysis of the Internet protocol TLS. *j-TISSEC*, 2(3):332–351, August 1999. (Cited on page 42.)

[288] Vern Paxson. Bro: a system for detecting network intruders in real-time. *Comput. Netw.*, 31(23-24):2435–2463, December 1999. (Cited on page 84.)

[289] Tao Peng, Christopher Leckie, and Kotagiri Ramamohanarao. Proactively detecting distributed denial of service attacks using source ip address monitoring. In *In Proceedings of the Third International IFIP-TC6 Networking Conference (Networking 2004*, pages 771–782. Springer, 2004. (Cited on page 85.)

[290] Tao Peng, Christopher Leckie, and Kotagiri Ramamohanarao. Survey of network-based defense mechanisms countering the dos and ddos problems. *ACM Comput. Surv.*, 39(1):3, 2007. (Cited on page 79.)

[291] A. Perrig and D. Song. Looking for diamonds in the desert - extending automatic protocol generation to three-party authentication and key agreement protocols. In *Proceedings of the 13th IEEE workshop on Computer Security Foundations*, pages 64–, Washington, DC, USA, 2000. IEEE Computer Society. (Cited on pages 44 and 45.)

[292] Adrian Perrig and Dawn Song. Looking for diamonds in the desert - extending automatic protocol generation to three-party authentication and key agreement protocols. In *In 13th IEEE Computer Security Foundations Workshop (CSFW*, pages 64–76. Society Press, 2000. (Cited on page 12.)

[293] Frédéric Perriot and Peter Szor. An analysis of the slapper worm exploit. Symantec Security Response. (Cited on page 19.)

[294] Carl Adam Petri. Kommunikation mit automaten., 1966. PhD thesis, Bonn: Institut fÃ¼r Instrumentelle Mathematik, Schriften des IIM Nr. 2, 1962. Second Edition:, New York: Griffiss Air Force Base, Technical Report RADC-TR-65â377, Vol.1, 1966, Pages: Suppl. 1, English translation. (Cited on page 6.)

[295] S. Petridou, S. Basagiannis, N. Alexiou, G. Papadimitriou, and P. Katsaros. Quantitative model checking of an RSA-based email protocol on mobile devices. In *Proc. 16th IEEE Symposium on Computers and Communications (ISCC'11)*, 2011. (Cited on page 37.)

[296] Joost pieter Katoen, Maneesh Khattri, and Ivan S. Zapreev. A markov reward model checker. In *in QESTâ05, Proc. 2nd Intl. Conf. on the Quantitative Evaluation of Systems*, pages 243–244. IEEE CS Press, 2005. (Cited on page 37.)

[297] Juan Carlos Lopez Pimentel, Raul Monroy, and Dieter Hutter. A method for patching interleaving-replay attacks in faulty security protocols. *Electron. Notes Theor. Comput. Sci.*, 174:117–130, May 2007. (Cited on pages 12 and 45.)

[298] PRISM. PRISM: Probabilistic Symbolic Model Checker. http://www.prismmodelchecker.org/. (Cited on page 37.)

[299] R. Ramanujam and S. P. Suresh. Tagging makes secrecy decidable with unbounded nonces as well. In *Proceedings, Foundations of Software Technology and Theoretical Computer Science (FST TCS 2003), volume 2914 of Lecture Notes in Computer Science*, pages 363–374. Springer, 2003. (Cited on page 47.)

[300] R. Ramanujam and S. P. Suresh. Decidability of context-explicit security protocols. *Journal of Computer Security*, 13:2005, 2005. (Cited on page 47.)

[301] Francesco Ranzato. On the completeness of model checking. In *Proc. 10 th ESOP ' 2001 , Genova, IT, 2–6 Apr. 2001, LNCS 2028*, pages 137–154. Springer-Verlag, 2001. (Cited on page 47.)

[302] Kavita Ravi and Fabio Somenzi. High-density reachability analysis. In *ICCAD '95: Proceedings of the 1995 IEEE/ACM international conference on Computer-aided design*, pages 154–158, Washington, DC, USA, 1995. IEEE Computer Society. (Cited on page 33.)

[303] Robert Richardson. 2008 csi computer crime & security survey, 2008. (Cited on page 78.)

[304] Robert Richardson. 2010/2011 csi computer crime and security survey, 2011. (Cited on page 78.)

[305] R. L. Rivest, A. Shamir, and D. A. Wagner. Time-lock puzzles and timed-release crypto. Technical report, Cambridge, MA, USA, 1996. (Cited on pages 88 and 94.)

[306] Martin Roesch. Snort - lightweight intrusion detection for networks. In *Proceedings of the 13th USENIX conference on System administration*, LISA '99, pages 229–238, Berkeley, CA, USA, 1999. USENIX Association. (Cited on page 84.)

[307] A. W. Roscoe, C. A. R. Hoare, and Richard Bird. *The Theory and Practice of Concurrency*. Prentice Hall PTR, Upper Saddle River, NJ, USA, 1997. (Cited on pages 6 and 28.)

[308] J. Rosenberg, H. Schulzrinne, G. Camarillo, A. Johnston J. Peterson, R. Sparks, M. Handley, and E. Schooler. Sip: Session initiation protocol. RFC 3261, June 2002. http://www.ietf.org/rfc/rfc3261.txt. (Cited on page 80.)

[309] Michal Rusinowitch and Mathieu Turuani. Protocol insecurity with finite number of sessions is np-complete. In *Theoretical Computer Science*, pages 174–190, 2001. (Cited on page 27.)

[310] Peter Ryan, Steve Schneider, Michael Goldsmith, Gavin Lowe, and Bill Roscoe. *The Modeling and Analysis of Security protocols*. Addison Wesley, 2000. (Cited on pages 15, 23, 26, 28 and 49.)

[311] Seungwan Ryu, Christopher Rump, and Chunming Qiao. Advances in active queue management (aqm) based tcp congestion control. *Telecommunication Systems*, pages 317–351, 2004. (Cited on page 108.)

[312] Dijkstra S. Modeling active queue management algorithms using stochastic petri nets. Faculty of Electrical Engineering, Mathematics and Computer Science, University of Twente, 2004; 79. (Cited on pages 3, 107 and 116.)

[313] Hassen Saidi. Towards automatic synthesis of security protocols. March 2002. (Cited on page 44.)

[314] M. Satyanarayanan. Integrating security in a large distributed system. *ACM Trans. Comput. Syst.*, 7(3):247–280, 1989. (Cited on page 23.)

[315] Stefan Savage, David Wetherall, Anna Karlin, and Tom Anderson. Practical network support for ip traceback. *SIGCOMM Comput. Commun. Rev.*, 30(4):295–306, August 2000. (Cited on page 85.)

[316] Steve Schneider. Security properties and csp. In *Proceedings of the 1996 IEEE Symposium on Security and Privacy*, SP '96, pages 174–, Washington, DC, USA, 1996. IEEE Computer Society. (Cited on page 15.)

[317] Koushik Sen, Mahesh Viswanathan, and Gul Agha. Statistical model checking of black-box probabilistic systems. In *In 16th conference on Computer Aided Verification (CAVâ04), volume 3114 of LNCS*, pages 202–215. Springer, 2004. (Cited on page 37.)

[318] Koushik Sen, Mahesh Viswanathan, and Gul Agha. On statistical model checking of stochastic systems. In *In Etessami, K., Rajamani, S.K., eds.: CAV. Volume 3576 of Lecture Notes in Computer Science*, pages 266–280. Springer, 2005. (Cited on page 37.)

[319] Koushik Sen, Mahesh Viswanathan, and Gul Agha. On statistical model checking of stochastic systems. In *In Etessami, K., Rajamani, S.K., eds.: CAV. Volume 3576 of Lecture Notes in Computer Science*, pages 266–280. Springer, 2005. (Cited on pages 100 and 101.)

[320] Michele Bugliesi Silvia, Silvia Crafa, Amela Prelic, and Vladimiro Sassone. Secrecy in untrusted networks. In *In ICALP â03*, pages 969–983. Springer, 2003. (Cited on page 39.)

[321] D. SISALEM, S. EHLERT, D. GENEIATAKIS, G. KAMBOURAKIS, T. DAGIUKLAS, J. MARKL, M. ROKOS, O. BOTRON, J. RODRIGUEZ, and J. AND LIU. Towards a secure and reliable voip infrastructure. Towards a secure and reliable VoIP infrastructure. Tech. rep. D2.1. SNOCER. May 2005. (Cited on page 81.)

[322] Dorgham Sisalem, Jiri Kuthan, and Günter Schäfer. Denial of service attacks and sip infrastructure: Attack scenarios and prevention mechanisms. 2005. (Cited on page 81.)

[323] J. Smith, J. M. González-Nieto, and C. Boyd. Modelling denial of service attacks on jfk with meadows's cost-based framework. In *ACSW Frontiers '06: Proceedings of the 2006 Australasian workshops on Grid computing and e-research*, pages 125–134, Darlinghurst, Australia, Australia, 2006. Australian Computer Society, Inc. (Cited on pages 97, 101, 129 and 131.)

[324] Jason Smith, Suratose Tritilanunt, Colin Boyd, Juan M. Gonzalez Nieto, and Ernest Foo. Denial of service resistance in key establishment. *Int. J. Wire. Mob. Comput.*, 2(1):59–71, May 2007. (Cited on pages 87, 92, 94 and 96.)

[325] Alex C. Snoeren, Craig Partridge, Luis A. Sanchez, Christine E. Jones, Fabrice Tchakountio, Stephen T. Kent, and W. Timothy Strayer. Hash-based ip traceback. *SIGCOMM Comput. Commun. Rev.*, 31(4):3–14, August 2001. (Cited on page 85.)

[326] Dawn Xiaodong Song, Sergey Berezin, and Adrian Perrig. Athena: A novel approach to efficient automatic security protocol analysis. *Journal of Computer Security*, 9(1/2):47–74, 2001. (Cited on pages 12, 35 and 36.)

[327] Graham Steel and Alan Bundy. Attacking group protocols by refuting incorrect inductive conjectures. *J. Autom. Reason.*, 36(1-2):149–176, 2006. (Cited on page 41.)

[328] Graham Steel, Alan Bundy, and Ewen Denney. Finding counterexamples to inductive conjectures. *AISB Journal*, 1, 2002. (Cited on pages 12 and 41.)

[329] Ling Su, Rong Zheng, and J.C. Hou. An active queue management scheme for internet congestion control and its application to differentiated services. In *Computer Communications and Networks, 2000. Proceedings. Ninth International Conference on*, pages 62 –68, 2000. (Cited on page 107.)

[330] Jinsheng Sun and Moshe Zukerman. Raq: A robust active queue management scheme based on rate and queue length. *Comput. Commun.*, 30(8):1731–1741, June 2007. (Cited on page 108.)

[331] Paul Syverson. A taxonomy of replay attacks. In *In Proceedings of the 7th IEEE Computer Security Foundations Workshop*, pages 187–191. Society Press, 1994. (Cited on page 46.)

[332] Paul Syverson, Catherine Meadows, and Iliano Cervesato. Dolev-yao is no better than machiavelli. In *First Workshop on Issues in the Theory of Security â WITSâ00*, pages 87–92, 2000. (Cited on page 27.)

[333] Paul F. Syverson and Iliano Cervesato. The logic of authentication protocols. In *FOSAD '00: Revised versions of lectures given during the IFIP WG 1.7 International School on Foundations of Security Analysis and Design on Foundations of Security Analysis and Design*, pages 63–136, London, UK, 2001. Springer-Verlag. (Cited on pages 18 and 45.)

[334] A. Neidhardt T. V. Lakshman and T. J. Ott. The drop-from-front strategy in tcp over atm and its interworking with other control features. in IEEE INFOCOM. (Cited on page 107.)

[335] AVANTSSAR Team. http://www.avantssar.eu/index.php?option=com_frontpage&Itemid=1. (Cited on page 50.)

[336] The AVISPA Team. AVISPA v1.1 User Manual. Document Version 1.1, June 30, 2006
www.avispa-project.org. (Cited on page 12.)

[337] Alwen Tiu. A trace based bisimulation for the spi-calculus: an extended abstract. In *Proceedings of APLAS 2007*, volume LNCS 4807, pages 367–382. Copyright Springer-Verlag.©, 2007. (Cited on page 38.)

[338] Simona Orzan Jun Pang Tom Chothia and Mohammad Torabi Dashti. A framework for automatically checking anonymity with μcrl. In *Trustworthy Global Computing*, volume 4661 of *Lecture Notes in Computer Science*, pages 301–318. Springer, 2007. (Cited on page 23.)

[339] Suratose Tritilanunt. Using coloured petri nets to simulate dos-resistant protocols, 2006. Proceedings of the seventh workshop and tutorial on practical use of coloured petri nets and the CPN tools. Denmark: University of Aarhus. (Cited on pages 97, 100 and 101.)

[340] Suratose Tritilanunt. Protocol engineering for protection against denial-of-service attacks, 2009. (Cited on pages 50, 92 and 96.)

[341] Suratose Tritilanunt, Colin Boyd, Ernest Foo, and Juan Manuel González Nieto. Cost-based framework and simulation of dos-resistant protocols using coloured petri nets. In *ACSC*, pages 191–200, 2007. (Cited on pages 97 and 118.)

[342] Suratose Tritilanunt, Colin Boyd, Ernest Foo, and Juan Manuel González Nieto. Toward non-parallelizable client puzzles. In *Proceedings of the 6th international conference on Cryptology and network security*, CANS'07, pages 247–264, Berlin, Heidelberg, 2007. Springer-Verlag. (Cited on pages 87, 91 and 92.)

[343] Randal Vaughn and Gadi Evron. Dns amplification attacks, 2006. available on www.isotf.org/news/DNS-Amplification-Attacks.pdf. (Cited on pages 77 and 82.)

[344] W. Eddy Verizon. Tcp syn flooding attacks and common mitigations. RFC 4987, 2007, URL: http://tools.ietf.org/html/rfc4987♯section-3.6. (Cited on page 80.)

[345] Luis von Ahn, Manuel Blum, Nicholas J. Hopper, and John Langford. Captcha: Using hard ai problems for security. In *IN PROCEEDINGS OF EUROCRYPT*, pages 294–311. Springer-Verlag, 2003. (Cited on page 82.)

[346] Huabing Yang; Xingyuan Zhang; Yuanyuan Wang. A correctness proof of the srp protocol. In *Parallel and Distributed Processing Symposium, 2006. IPDPS 2006. 20th International*, 2006. (Cited on page 43.)

[347] Brent Waters, Ari Juels, J. Alex Halderman, and Edward W. Felten. New client puzzle outsourcing techniques for dos resistance. In *CCS '04: Proceedings of the 11th ACM conference on Computer and communications security*, pages 246–256, New York, NY, USA, 2004. ACM. (Cited on page 89.)

[348] Christoph Weidenbach. Towards an automatic analysis of security protocols in first-order logic. In *Conference on Automated Deduction*, pages 314–328, 1999. (Cited on page 40.)

[349] Christoph Weidenbach and Im Stadtwald. Spass: Combining superposition, sorts and splitting, 1999. (Cited on pages 40 and 41.)

[350] Huabing Yang Xingyuan Zhang and Yuanyuan Wang. Liveness reasoning for inductive protocol verification. In *In The 'Emerging Trend' of TPHOLs 2005, Oxford University Computing Lab. PRG-RR-05-02*, pages 221–235,. (Cited on page 43.)

[351] Haifeng Xue, Huanguo Zhang, and Sihan Qing. A schema of automated design security protocols. In *Proceedings of the 2007 International Conference on Computational Intelligence and Security Workshops*, CISW '07, pages 733–736, Washington, DC, USA, 2007. IEEE Computer Society. (Cited on page 45.)

[352] Abraham Yaar, Adrian Perrig, and Dawn Xiaodong Song. Siff: A stateless internet flow filter to mitigate ddos flooding attacks. In *IEEE Symposium on Security and Privacy*, pages 130–, 2004. (Cited on page 86.)

[353] Xiaowei Yang, David Wetherall, and Thomas Anderson. A dos-limiting network architecture. In *Proceedings of the 2005 conference on Applications, technologies, architectures, and protocols for computer communications*, SIGCOMM '05, pages 241–252, New York, NY, USA, 2005. ACM. (Cited on page 86.)

[354] Yannick Chevalier and Laurent Vigneron. (Cited on page 31.)

[355] Hakan L. S. Younes. Ymer: A statistical model checker. In *COMPUTER AIDED VERIFICATION. LNCS*, pages 429–433. Springer, 2005. (Cited on page 37.)

[356] Changwang Zhang, Jianping Yin, Zhiping Cai, and Weifeng Chen. Rred: robust red algorithm to counter low-rate denial-of-service attacks. *Communications Letters, IEEE*, 14(5):489 –491, may 2010. (Cited on page 108.)

[357] Changwang Zhang, Jianping Yin, Zhiping Cai, and Weifeng Chen. Rred: robust red algorithm to counter low-rate denial-of-service attacks. *Communications Letters, IEEE*, 14(5):489 –491, may 2010. (Cited on page 109.)

[358] Ninggrong Zhong, Xingyuan Zhang, and Yuanyuan Wang. Formal analysis of gm multi-party contract signing protocol. In *ICCIT '07: Proceedings of the 2007 International Conference on Convergence Information Technology*, pages 1316–1321, Washington, DC, USA, 2007. IEEE Computer Society. (Cited on page 43.)

[359] Hongbin Zhou and Simon N. Foley. Fast automatic synthesis of security protocols using backward search, 2003. (Cited on page 44.)

[360] Andre Zuquete. Improving the functionality of syn cookies. In *Proceedings of the IFIP TC6/TC11 Sixth Joint Working Conference on Communications and Multimedia Security: Advanced Communications and Multimedia Security*, pages 57–77, Deventer, The Netherlands, The Netherlands, 2002. Kluwer, B.V. (Cited on page 106.)