NTNU
Norwegian University of
Science and Technology

# Authentication in Protected Core Networking

Kristian Malmkvist Eie

Master of Science in Communication Technology

| | |
|---|---|
| Submission date: | June 2016 |
| Supervisor: | Danilo Gligoroski, ITEM |
| Co-supervisor: | Anne Marie Hegland, Kongsberg Defence and Aerospace |

Norwegian University of Science and Technology
Department of Telematics

| | |
|---|---|
| **Title:** | Authentication in Protected Core Networking |
| **Student:** | Kristian Malmkvist Eie |

**Problem description:**

Modern military operations are getting more cooperative, and with that, more complex in terms of information sharing. As a result, the concept of network enabled capabilities (NEC) has been formed to create information superiority by adapting faster to changing situations and improving decision making. In this setting, communication is a basic challenge. Protected Core Networking (PCN) is a fairly new concept intended to be used to implement a secure and flexible transport communications infrastructure that supports the requirements of NEC operations. PCN specifies a loose coupling between information domains and the transport networks. The users, and information domains are located in Colored Clouds (CCs) that are separated from the protected core (PCore) transport network. Only network elements that contribute to the transport of information are part of the PCore. Whereas PCN provides a protected network service, it does not provide confidentiality protection of user data. This must be provided by the CCs. The main goal of the PCore is to provide high-availability transport services to its users. The PCore must ensure that the service level agreements (SLAs) are met. Access control and mitigation of denial of service attacks are vital parts of that. As preventive measures, PCN specifies initial entity authentication of all connecting entities, and message authentication of all datagrams sent onto links in the PCore. However, detailed specifications for the authentication solution lack, and evaluation of different alternatives is still to be provided. Focusing on authentication in PCN, the intention of this thesis is to fill at least part of this gap.

A much used form of authentication is proving possession of a key. Regardless of whether symmetric or asymmetric keys are used, the verifying entity is required to be in possession of a verification key. Although certificate-based PKIs are widely deployed in practice today, the distribution of certificates is often complicated. In 1984 Shamir introduced identity-based cryptography where any unique string could be used as the public key, and where certificates are superfluous. Typically, IDs are used as public keys. A message signed with the private key can then be verified by any peer using the ID of the originator of the message. A main objective is to study the applicability of one or more identity-based cryptographic schemes for authentication in PCN. Even though identity-based cryptography is not without its drawbacks, and comes with challenges such as revocation and key escrow, it provides an option to traditional certificate-based PKIs. The goals of this thesis are to:

- Study relevant solutions for using identity-based cryptography in PCN authentication.

– Evaluate a solution, and analyze its security using formal methods.
– Discuss issues and challenges of deployment of the suggested solution.

| | |
|---|---|
| **Responsible professor:** | Danilo Gligoroski, ITEM NTNU |
| **Supervisor:** | Anne Marie Hegland, Kongsberg Defence and Aerospace |

## Abstract

Protected Core Networking (PCN) is a concept that aims to increase information sharing between nations in coalition military operations. PCN specifies the interconnection of national transport networks, called Protected Core Segments (PCSs), to a federated transport network called Protected Core (PCore). PCore is intended to deliver high availability differentiated transport services to its user networks, called Colored Clouds (CCs). To achieve this goal, entity authentication of all connecting entities is specified as a protective measure. In resource constrained environments, the distribution of service policy can be challenging. That is, which transport services are associated with a given entity. The thesis proposes two new and original protocols where CCs push service policy to the network by performing authentication based on attributes. Using identity-based signatures, attributes constituting a service policy are used directly for an entity's identity, and no external mechanism linking identity and policy is needed. For interoperability, the idea has been incorporated into PKINIT Kerberos and symmetric key Kerberos by carrying the attributes within tickets. The proposed protocols are formally verified in the symbolic model using scyther-proof. The experiment shows that both CCs, and PCSs achieve greater assurance on agreed attributes, and hence on expected service delivery. A CC and a visiting PCS are able to negotiate, and agree on the expected service depending on the situation. The proposed solution provides benefits to CCs on expected service when connecting to a visiting PCS, with poor connectivity to the home PCS. In that respect, interconnection of entities with little pre-established relationship is simplified, and hence fulfillment of the PCN concept is facilitated.

## Sammendrag

Protected Core Networking (PCN) er et konsept som har som mål å muliggjøre økt informasjonsdeling på tvers av nasjoner i militære koalisjonsdrevede operasjoner. PCN spesifiserer sammenkoblingen av nasjonale transportnettverk, kalt Protected Core Segments (PCSs), til et felles beskyttet transportnettverk, kalt Protected Core (PCore). PCore skal tilby dets brukernettverk, kalt Colored Clouds (CCs), høy tilgjengelighet av differensiert tjenestekvalitet. For å oppnå dette er autentisering av sammenkoblede enheter spesifisert som et beskyttende tiltak. I ressursfattige nettverk kan distribusjon av tjenestepolicy være en utfordring. Det vil si, hvilke transporttjenester som er assosiert med en entitet. I denne masteroppgaven foreslås to nye og originale protokoller der CC-er dytter tjenestepolicy til et PCS ved å utføre autentiseringen basert på attributter. Ved bruk av identitets-baserte signaturer kan attributter tilhørende en tjenestepolicy benyttes direkte som en CCs identitet, og ingen ekstern binding mellom identitet og policy er nødvendig. For interoperabilitet har ideen blitt benyttet i PKINIT-Kerberos og symmetrisk nøkkel-Kerberos ved å frakte attributter i *tickets*. Den foreslåtte løsningen er formelt verifisert i den symbolske modellen ved hjelp av scyther-proof. Eksperimentet viser at både CC og PCS oppnår større forsikring om enigheten av attributter, og dermed også på forventet tjenestelevering. En CC og et PCS er i stand til å forhandle om policy, og kan komme til enighet om forventet tjenestelevering avhengig av situasjonen. Den foreslåtte løsningen tilbyr fordeler for CC-er på forventet tjenestelevering når den kobler seg til et fremmed PCS, med lav konnektivitet til dets hjemlige PCS. På denne måten forenkles sammenkoblingen av enheter med lite forhåndsdefinert forhold, noe som understøtter målet med PCN.

# Preface

The thesis was written as part of the master's degree in communications technology, at the institute of telematics, NTNU. The topic was suggested by Kongsberg Defence and Aerospace, which plays a part in the development of PCN. The idea of Network Enabled Capabilities (NEC) has been around for a while, and PCN suggests a way of enhancing the flexibility of communication at the lowest decision level, enabling NEC. In PCN, authentication is suggested as a way of ensuring availability of transport service through enforcement of access control on all connecting entities. The wide range of solutions of authentication motivated a study of protocols fit for PCN.

I would like to thank supervisors Anne Marie Hegland and Hans Are Ellingsrud at Kongsberg Defence and Aerospace for suggesting this topic, and much valued guidance on the topics of PCN and network security. I would also like to thank responsible professor Danilo Gligoroski for guidance on academic writing and on the overall structure of the report. Finally, I am grateful for clarifications on the specifications of PCN given by Ingar Bernstuen, Per Carlén and Geir Hallingstad. A thank you to Cas Cremers for advice on scyther, and to Simon Meier on scyther-proof and Tamarin is also deserved. A special thanks to family and friends who have provided me with infinite support and understanding during these years.

Trondheim, June 2, 2016
Kristian Malmkvist Eie

# Contents

# List of Figures

# List of Tables

# Chapter 1 Introduction

Maybe you have heard it before: the world is getting smaller, more complex, and it changes faster than before. Humanity has probably been saying it for centuries. Nevertheless, it is probably not less true today. New alliances and bonds are formed continuously in this increasingly complex world. The essence of these bonds – trust – is believed by many to depend on good communications. With situations changing quickly, the ability to form new bonds and communicate effectively is a valued trait. Cooperative work can only be effective with the right information, at the right time. Recently, effective information sharing in military operations has become increasingly difficult. In a much larger degree than previously, military operations are now often coalition operations where parties cooperate to achieve common goals. Local governments, humanitarian organizations, and international forces create new trust relationships in order to work together. The ability to adapt faster to changing situations has become key to successful operations.

In military operations, increasingly more decisions are made at the lowest decision level because the command and control model is changing, moving towards edge-organizations [HO08]. Such a shift puts additional pressure on communications systems to make information available to decisions makers at the periphery of the network topology. To embrace these changes, the concept of Protected Core Networking (PCN) was proposed. In traditional military Communication and Information Systems (CISs), confidentiality services are tailored to the individual communications infrastructure [HO08]. The disadvantage of such an approach is that only information of the correct classification can be sent on a specific link. While such an approach may be acceptable in a single party engagement, it is clearly inefficient when multiple parties are in the same geographical area. Each nation or organization would need individual links going in the same direction with their own tailored confidentiality measures. In their proposal for an architecture of PCN [HO08], Hallingstad et.al. suggested that confidentiality services are eliminated from the transport network. Each nation or organization would have to apply confidentiality to its data at a higher protocol layer if needed. The network would then offer only service guarantees and timely delivery of data to its users. In essence, PCN creates a trusted overlay that controls service access. In order to offer dependable transport services to its users, threats such as network congestion, and malicious flooding of network links must be addressed. Quality of Service (QoS) is the functionality required for managing network resources in order to assure agreed service levels [SMD+13]. A common way of enforcing QoS is through authentication. The focus of this thesis is related to authentication, and specifically how Identity-based Public Key Cryptography (IDPKC) may be used in PCN. IDPKC will be defined shortly, but a brief introduction to the essentials of PCN will be given first.

## 1.1 Background

The concept of PCN was first outlined by Geir Hallingstad and Sander Oudkerk in their article from 2008 [HO08]. The concept has further been developed by the NATO Research and Technology Organization PCN group which have created Requirements for an Interoperability Specification (ISpec) for PCN [PCN10]. Selected challenges for protected core networking is discussed in [PCN14]. Furthermore, QoS in PCN is treated in [SMD+13]. These documents constitute the thesis' background on PCN.

A high level overview of PCN as described in [HO08] is included in Figure 1.1. The nodes marked 'E' in the figure are specialized routers with enforcements capabilities. That is, they are capable of enforcing access control on which packets are forwarded through the network. E-nodes are interconnected to form independent logical segments based on PCN capabilities called Protected Core Segments (PCSs). A PCS is interconnected, and cooperates, with other PCSs over interfaces, called PCN-1, to create a federated Protected Core (PCore) transport network. In a deployment based on PCN principles, each nation or organization could contribute with, and manage their own PCS. There is no central authority in a PCore as a whole, and the PCSs are interconnected in an ad hoc manner. Only network elements that contribute to the transport of information are part of PCore. The users, and information domains are located in Colored Clouds (CCs), which are Local Area Networks (LANs) set up by nations or other logical groups. The CCs are separated from PCore by encryption devices, marked with Z in Figure 1.1. CCs will connect to PCore over interfaces called PCN-2, which will be further explained shortly. The CCs in Figure 1.1 are drawn with different shades of grey to emphasize that user networks with information of different classifications can utilize PCore. Similarly, the PCore itself is drawn white to symbolize that information is treated similarly regardless of classification.

It is emphasized that PCN is not concerned with the internal implementations of PCSs. Rather, that the heterogeneous networks implemented as PCSs delivery services to other PCSs and CCs according to PCN-1 and PCN-2 interfaces. The main focus of the thesis will be on PCN-2 interface. As mentioned briefly above,

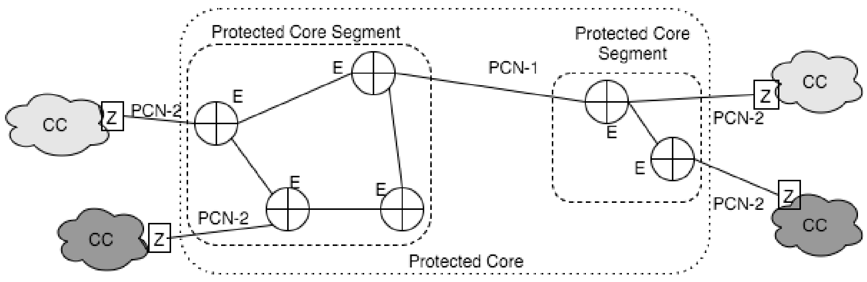

**Figure 1.1:** Overview of PCN as described in [HO08]. User networks holding information are located in CCs. CCs are in turn connected to PCore through PCN-2.

PCN-2 defines a loose coupling between a CC and PCore. An important requirement is that CCs should be able to move geographically and reconnect to PCore elsewhere, without manual reconfiguration [PCN10].

To support different services, such as voice or data transfer, the PCore is intended to have extensive support for QoS enforcement. The fact that critical traffic should take priority over less important traffic is highlighted in [HO08]. A Service Level Agreement (SLA) between a CC and a PCS is negotiated, or pre-stored, in order to determine which services will be provided and how they will be delivered to a CC [SMD+13]. Its purpose is to give CCs the ability to predict which service can be expected from the PCS. Similarly, it gives PCSs the ability to prepare to deliver agreed service. A SLA can include parameters such as priority, bandwidth, and delay depending on what is required for a given service. To verify the identity of a connecting entity correctly, entity authentication of all connecting entities is required [PCN10]. Additionally, individual packets travelling the PCore are required to be protected using transaction authentication. These terms, entity authentication and transaction authentication, will be informally defined shortly.

## 1.2 Terminology

Terminology slightly modified from [HWH11] are stated here for further reference.

**Identity and identifiers:** An Identity (ID) uniquely specifies an entity. An identity is represented by one or more identifiers. An entity can be a single CC or E-node, or a group of these referred to by one identifier. For example, CCs can have individually distinguishable identities, but being part of a PCS, they can share an identity separate from their individual ones.

**Entity authentication:** The process, or protocol, where one entity verifies that another entity communicates with a claimed identity. After the identity is verified, the process ends. Entity authentication can be mutual or one-way. Often there is also a notion of time involved, i.e. guarantees on the recentness of authentication. Several different forms of entity authentication will be defined in chapter 3. The main difference amongst them being the assurance one entity has on the state of the other entity after authentication.

**Authorization:** The act of determining which services an identified entity should have access to is called authorization. As authorization depends on identification, the decision should be made based on authenticated identifiers. A specific identity can be authorized for some service, or a user is authorized based on some identifier shared between multiple users, such as roles or attributes.

**Attribute:** An attribute is a quality or feature inherent in some entity. For a CC, an attribute might be some property such as the type of vehicle or ship, or which nation it is from. A service attribute is defined as an attribute with a specific set of services attached. For example, a reconnaissance ship might have certain predefined services attached to it, such as continued connectivity, or limitations on delay.

**Transaction authentication:** Data-origin authentication with time variant parameters is often called transaction authentication. The receiver is assured that the message originated from the claimed source within a given time interval.

Having stated informal definitions of central concepts above, these will be used when outlying challenges of doing authentication in distributed systems below.

## 1.3 Objectives

Separating entity authentication from transaction authentication can be advantageous in distributed systems [EA04]. Entity authentication can then be performed by a central server for Authentication, Authorization, and Accounting (AAA), while access control of packets is enforced on distributed routers. Before transaction authentication is applied to packets, the entity enforcing access control must have verified the source identity so as to know which packets to forward. In PCN, certain transport services might also be associated with an identity, i.e. the service policy. Maintenance and distribution of service policy is often simplified through centralized control. To achieve a flexible approach to QoS enforcement, the policy is here assumed dynamic. That is, entities such as AAA-servers, E-nodes, and CCs may all be involved in the process of determining the quality of services delivered. The goal is better utilization of network resources. To achieve this, one challenge is the transportation of service policy from a CC's home PCS to visiting PCS. Another is how policy is transported from a central server to distributed E-nodes. These two challenges are addressed in this thesis.

Entity authentication is often performed by proving knowledge of a key. In public key cryptography, a private key is linked to a public key through a one-way trapdoor function. A one-way trapdoor function is one that is easy to compute, but difficult to invert without a secret key. This corresponds to (1.3) in Figure 1.2. The public key is in turn usually bound to the owner's identity (1.2), which finally is linked to a service policy (1.1). To assure (1.2) a trusted entity, called Certificate Authority (CA), is often assumed to issue certificates to other entities. A certificate is a document stating that a CA has verified the binding between an identity and a public key. In Public Key Infrastructures (PKIs) certificates are used to create tree structures where leaf nodes only trust the root. Then, if entity $CC$ trusts $CA$, and $CA$ trusts $E$, then transitively $CC$ trusts $E$. Nevertheless, if a CC's public key, identity, and policy are known to an E-node in advance, the authenticity of these bindings are assured by secure physical storage. A CC would then only prove knowledge of the private key to an E-node in order to be linked with a service policy. However, pre-storing this information in all E-nodes for large number of CCs does not scale well.

In 1985 Shamir introduced Identity-based Public Key Cryptography (IDPKC) [Sha85]. The idea is that *any* chosen string could be used as a public key. By choosing this string to be an identity, the public key is essentially the same as the identity. No external mechanism such as certificates, is then needed to ensure the binding between the two (2.2). In IDPKC, a Key Generation Center (KGC) is involved in the creation of private keys. Part of this process, entities are authenticated by the KGC when receiving private keys. This process is referred to as pre-authentication. Pre-authentication gives verifying entities assurance that a KGC has authenticated a peer's identity before issuing a private key based on their identity. The overarching objectives of the thesis are to:

- Study relevant solutions for using IDPKC in PCN authentication
- Evaluate a solution, and analyze its security using formal methods
- Discuss issues and challenges of deployment of the suggested solution

Even though authentication may be simplified, IDPKC does not immediately solve the challenge of distributing service policy to E-nodes. An underlying objective is to examine the possibility of including information within an entity's identity; enabling distributed authentication by simplifying row (2) of Figure 1.2 even further.

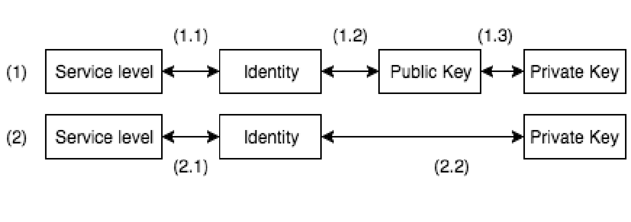

**Figure 1.2:** A comparison of classical public key cryptography (1) to IDPKC (2).

## 1.4 Methodology

Formal methods are ways of using mathematics and logical notations to describe a protocol. By formally stating security goals protocols need to achieve, models can be run against an attacker's assumed capabilities to evaluate its robustness. Formal methods can detect wrong assumptions, an incorrect protocol, and model if principals leak information. There are essentially two ways of providing formal security guarantees to cryptographic protocols [BM13]. One way is through complexity reductionist proofs that reduces the hardness of an unknown problem to that of a known problem, and hence builds its security on those reductions and the base problem. Then there are methods that treat cryptographic mechanisms as black boxes with perfect security, called *symbolic models*. The approach taken by the thesis is of the latter category.

Generally, one can divide formal methods of symbolic analysis into those that aim to prove a protocol correct, and those that aim at finding flaws. Where ProVerif [Bla01] and scyther-proof [Mei13] are examples of the former, Automated Validation of Internet Security Protocols and Applications (AVISPA) [ABB$^+$05] and scyther [Cre06] represent the latter. Scyther will here be used for preliminary analysis of protocols. Furthermore, a theorem proving tool built on scyther, called scyther-proof, will be used to prove security properties correct. These tools will be explained in more detail in chapter 3.

## 1.5 Limitations

All entities are assumed to have global system parameters pre-stored in a tamper resistant module before deployment. When CCs are moving geographically and reconnecting to PCore through other PCSs, global parameters for several PCSs are needed. An authority in each security domain is assumed to generate private keys for entities before deployment. The process of deploying keying material in advance is assumed not compromised. Figure 1.3 shows that trust relationships between PCSs must exist before deployment in order allow interconnection. Furthermore, mechanisms for rekeying of private keys are not analyzed in this thesis. This choice is justified by not assuming compromising adversaries. If PCN is adopted to the tactical domain, this assumption may be too strong. It is also important to note the inherent limitations of formal methods in a symbolic model. The attacker model and security goals are critical in evaluating the security of the protocol. Formal methods will not detect errors such as wrong implementation, or wrong usage. In the symbolic model, cryptographic primitives are considered to have perfect security, and hence are not considered in the model.

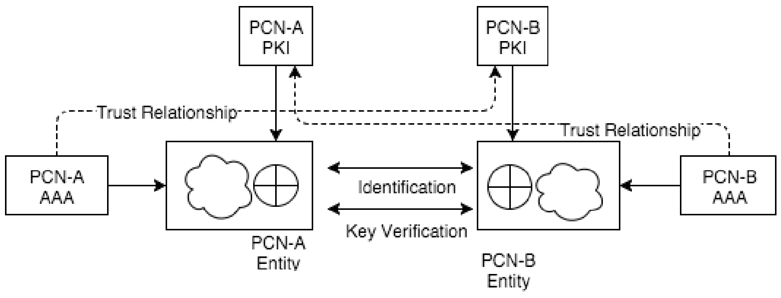

**Figure 1.3:** Figure from [PCN10] showing that a trust relationship between connecting PCSs must exist prior to authentication

## 1.6 Outline

Following is an outline of the content of this thesis. This first chapter was intended to introduce the concept of PCN, and to motivate the study of IDPKC for authentication in PCN.

Chapter 2 initially states high level requirements for mechanisms supporting authentication in PCN. It further looks at the PCN architecture from a high level perspective, and briefly discuss id-based cryptography in Mobile Ad Hoc Networks (MANETs), the topic of distributed authentication, and relevant Internet standards.

Chapter 3 explains concepts, and more theoretical foundations for many for the mechanisms used throughout the thesis. Specifically, it gives background information on IDPKC, and formal methods for verification of cryptographic protocols which extend upon section 1.4 in this chapter.

Having laid the ground work in chapters 2 and 3, chapter 4 propose solutions, and outlines the details through two new protocols. The contents of chapter 5 is the experimental part of the thesis where the proposed protocols are formally analyzed in a tool called scyther-proof. An adversary model is defined, and subsequently so are security goals before the analysis. Results obtained are provided at the end of the chapter.

Chapter 6 discuss the results from chapter 5 with respect to the adversary model and security goal. In addition, it contains a discussion of issues related to a practical deployment of the proposed solution, with a specific focus on identity-based cryptography and signature schemes.

Chapter 7 summarizes the thesis by looking at assumptions, modelling, and results. It also tries to answer the question if the objectives stated in section 1.3 were fulfilled. Finally, topics for further work are stated in section 7.2.

Chapter 2

# A literature survey on authentication and IDPKC for potential use in PCN

The goal of this chapter is to study relevant literature of authentication and IDPKC for potential use in PCN. A high level overview of different solutions is given in order to identify missing pieces of existing research. Focus will be on distributed systems in environments with limited bandwidth.

Shortly a few requirements will be stated which will aid in the search for a viable solution. What is meant by the *security*-requirement is the assurance a CC and an E-node has on attributes defining expected service delivery. That is, to what extent they have the same view of the authentication protocol and on agreed data items. Later, more accurate definitions of authentication, such as Lowe's [Low97] definitions of agreement, will be applied to dissect the problem more carefully. Figure 2.1 illustrates how the specifications and requirements for PCN, such as those related to authentication, are specified at the network layer. The thesis will follow in the same line in order for authentication mechanisms to be interoperable independent of selected access mediums.

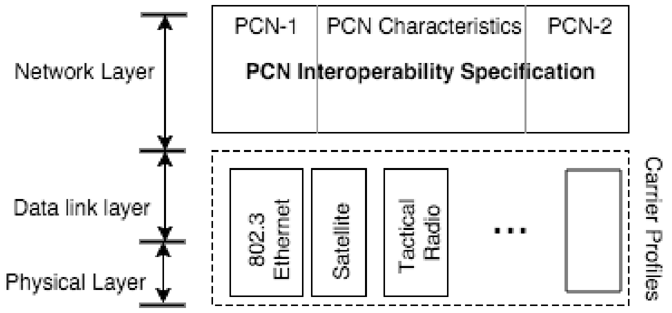

**Figure 2.1:** Figure showing how services in PCN, such as authentication, must run over different carrier profiles [PCN10].

## 2.1 Requirements

A few high level requirements found in PCN literature are stated here in order to make architectural decisions regarding authentication later in the chapter:

- Interoperable
- Scalable
- Efficient
- Secure

**Interoperability:** To avoid complex solutions when interconnecting heterogeneous networks, an important requirement for PCN is interoperability. Many technologies for the transportation of information exist. Examples include radio links, IEEE 802.11, IEEE 802.3, satellite, and WiMAX. In order to utilize several technologies for physical transportation and medium access, the thesis will be concerned with solutions enforcing transaction authentication at the network layer of the Open Systems Interconnection (OSI) stack.

**Scalability:** Scalability is concerned with the degree workload on central nodes increase with the size of the network. In a future deployment of PCN, large parts of national and international military entities may be connected to a common backbone. Scalability of operation in critical functionality, such as authentication mechanisms, is considered vital.

**Efficiency:** The many different communications technologies put constraints on solutions that is intended to run on top of a range of these. Efficiency of computations and bandwidth are important to achieve flexibility when deploying in operational scenarios. Stationary networks can also benefit from efficiency in the number of messages passes in connection setup traffic. Another goal is efficient interconnection of PCS networks, that is with minimal manual configuration.

**Security:** Security is here meant as providing the correct service to a given entity. It is important that the chosen solution provides guarantees to involved entities on authentication and secrecy goals in order to enforcement the correct services.

The above requirements will now be utilized when studying literature of authentication. It is important to note that fulfillment of these requirements must be balanced in distributed, heterogeneous networks.

## 2.2 Literature

When surveying the literature, focus was initially on IDPKC in Mobile Ad Hoc Networks (MANETs). Since MANETs are self-configuring, infrastructure-less networks, parallels can be drawn to the required interconnection of entities in PCN. Topics of distributed authentication and Internet standards were also covered.

### 2.2.1 Authentication in MANETs

In the following section two methods for doing transaction authentication are presented. Next, the trade-offs between public key, symmetric key, and IDPKC are discussed. Furthermore, the use of IDPKC in MANETs is motivated.

**Three level framework for authentication** In [HWH11] Hegland et.al. propose a three level framework for authentication in tactical ad hoc-networks in network based defense. They suggest network level hop-by-hop authentication as basic authentication of data-packets. Additionally, they suggest using group keys for symmetric authentication hop-by-hop, because one limits the scaling problems of using keys at the granularity of individuals. In addition, they reason that verification of group membership might in many applications be sufficient for basic authentication.

**Quality of service enforcement in Military Ad Hoc Networks** Military ad hoc networks are different from true ad hoc networks in that some planned structure can be assumed [HW08]. In order to fulfill QoS requirements in such a network Hegland et.al. propose a protocol for authenticating QoS fields in IP-header based on group keys. The resulting protocol is based on distributing group keys to specific intermediate nodes or routers which enables them to authenticate QoS fields. Identified drawbacks are insider attacks or node compromise, and the associated key distribution. One possible solution is the use of digital signatures and certificates to authenticate data packets. Hegland et.al. [HW08] briefly discuss this solution and mention that digital signatures might be a viable solution for fixed networks rather than tactical networks.

**Packet Level Authentication using Digital Signatures** In [CLK05] Candolin et.al. propose a generic solution for performing Packet Level Authentication (PLA) in heterogeneous military networks. They identify that a solution to security issues in a systems-of-systems approach could benefit from the simplicity of PLA. To support their claims, they propose a solution where each node in the network, and not only source and destination, can authenticate a data packet. Specifically, they suggest that an IP-packet is authenticated using digital signatures, and that a certificate is carried in the IP-extension header. While such an approach clearly simplifies key management, the computational overhead, and bandwidth consumption is significant.

**Strong authentication in MANETs** In [TD07] Tang et.al. discuss challenges of strong authentication in MANETs through the framework from [HG04]. Authentication using symmetric keys can be performed between two entities who share a key by proving possession of this key. One way of doing this is a classical challenge-response where one party takes the role of authenticator, and the other as supplicant. The authenticator encrypts a secret value, and asks the supplicant to decrypt. If mutual authentication is requested, the roles are changed. Advantages of symmetric key solutions are identified as efficiency of operations and computationally feasibility [TD07]. Typical issues are related to key distribution. Secret keys have to be securely deployed in each node. There is also an inherent trade-off between the number of entities who share a symmetric key and the security in case of key compromise. Some suggestions for reducing the number of key pairs are further discussed in [TD07], and includes for instance key hierarchies.

**Interoperability between national MANETs** In [SBW$^+$15] Salmanian et.al. propose an architecture for the interoperability between two national MANETs in a coalition. Part of their contribution is a key management scheme supporting interoperability between small platoon-sized MANETs from different nations. Their scheme utilizes a certificate-based PKI to simplify key distribution. Interestingly, they suggest using a special mission key for encrypting the header of a packet, while the payload is encrypted using a group key. To set up security associations between two nodes chosen as gateways, they use a combination of public key cryptography for distributing group keys, while using symmetric key cryptography for data confidentiality.

**Identity-based cryptography in MANETs** In [Tha15] Thakur and Heena provide a review on applications of IDPKC in MANETs. They survey literature on how IDPKC could be used to secure routing protocols in MANETs, and how it could be used in military settings. Specifically, the use of non-interactive key establishment, and pre-authentication are identified as major advantages of IDPKC. In [BBKP07], Balfe et.al. propose a scheme for using IDPKC in a military coalition structure. They build their scheme on that each security domain contains its own KGC issuing private key to entities in its domain. Disadvantages of IDPKC listed in [TD07] include key escrow, single point-of-failure, and the distribution of private keys. These will be discussed in chapter 6 in the context of PCN.

### 2.2.2 Authentication in distributed networks

The focus of the following section will be on authentication in distributed networks. Specifically, two solutions using the ticket-based architecture are covered.

**Authentication and the SESAME solution** Authentication can be particularly challenging in distributed federated networks due to the difficulty of synchronizing access rights. In [Ash97] Ashley et.al. use the framework of [CJW96] to examine the applicability of Secure European System for Applications in a Multivendor Environment (SESAME)[KPP94] to a multi-domain environment. They argue that it will often be the case that the foreign network does not have knowledge of the client in advance. Additionally, a system mapping access right, or attributes, across administrative domains are necessary. Finally, a trust system must be in place for a foreign network to trust the authenticity of a decision made by another network.

In [CJW96] Ching et.al. identified that traditional protection systems in closed environments intertwine the fields of naming, entity authentication, and access control. They suggest that in a large scale network, the access decision should not depend entirely on static identities and attributes. Information related to the environment from which the request originated, and local context were identified as other important parameters. In a multi-domain environment, it is rarely practical for a foreign network to store access policies for both foreign and local clients in a centralized database. For this reason, access rights are either *pulled* by the remote network server from the client's home network, or *pushed* by the client's application [Ash97]. The reader is referred to the article for the full discussion, but for example a push-based architecture normally creates less communications traffic, and a decision can be made more efficiently [Ash97]. Another advantage of the push-based architecture, is that the access decision can be tempered based how the client is authenticating, and on properties of the environment.

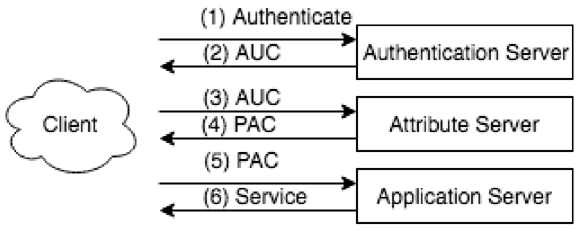

**Figure 2.2:** In SESAME [KPP94] an application server perform access control purely on attributes and local state, without knowing the client in advance.

Figure 2.2 illustrates the main entities involved. A client authenticates to an authentication server, and receives an Authentication Certificate (AUC) stating that the it has been authenticated. The client can in turn present the AUC to an attribute server, and receive back proof of its access rights in a special access control certificate, called Privilege Certificate (PAC). Finally, the PAC is presented by the client to an application server whenever access to a protected resource is needed. Based upon the attributes, and other local information, the application takes a decision. The PAC is thought to be a data structure protected using public key cryptography [KPP94].

**Federated authentication and ticket-based guest access** An Identity Management System (IdM) is a system that maintains information regarding actors in an information system. It provides services such as validation of credentials for the purpose of authentication. In [Fon10][FD11] Fongen identify that IdMs can bridge the gap between two security domains. Conforming with traditional user management, [FD11] stress that access rights should be linked to roles or attributes rather than directly to users. A central concept in order to achieve this is Identity Statements (ISs). The ISs are short lived service tickets to remove the need for Certificate Revocation Lists (CRLs). CRLs are lists maintained by PKIs on invalidated credentials. Specifically, an IS binds an identity to a public key, and a set of attributes. It is signed by an authority trusted by all entities within a domain. By having access control be performed on attributes, rather than identities, a loose coupling between information domains and services are achieved. A central concept in order to achieve this is *guest access*. With guest access, a user's home network issues a guest-ticket consisting of attributes associated with that identity. It is assumed that the home network has a pre-established trust relationship with the visiting network. Upon connecting to the visiting network, the user presents its guest-ticket to the network. The visiting network verifies the signature and then determines which services it associates with the presented attributes. A new foreign-guest-ticket is issued by the visiting network, possibly with new attributes, and digitally signed.

### 2.2.3 Interoperability in IP-networks

Internet Engineering Task Force (IETF) specifies open standards for protocols and services on the Internet. Internet Protocol Security (IPSec) provides services such as access control, and data origin authentication to IP-packets [KS05]. In order to keep track of which services will be applied to IP-packets shared state, called Security Associations (SAs), are maintained between source and sink of an IPSec connection [KHNE14]. Two standards for setting up SAs for IPSec are covered, namely Internet Key Exchange (IKE) and Kerberized Internet Negotiation of Keys (KINK). IKE is built on a Diffie-Hellman key exchange [DH76] authenticated with digital signatures. KINK on the other hand is based on Kerberos.

**Id-based Internet Key Exchange** IKE is cryptographically built from the Sigma protocols of authentication key exchange [Kra03]. In [SD03] Smetters et.al. identify that the extensibility of IKE, with negotiating parameters initially, would help almost seamlessly incorporate IDPKC into IKE. They argue that as long a IDPKC mechanisms can be used in existing protocol flows and message fields, few changes would be needed. When using IDPKC, certificates are superfluous because one can authenticate by proving possession of the private key corresponding to an identity. The identity itself can be known by the peer beforehand, or sent along in IKE exchange. In RFC 4306 IKEv2 [KHNE14], a number of possible formats for identities are specified, such as Fully Qualified Domain Name (FQDN) and IP-addresses. If IP-address are used as identities, [CNC$^+$04] conclude that the ID-field in Figure 2.3 could be eliminated. Appendix A provides additional information on the fields of Figure 2.3.

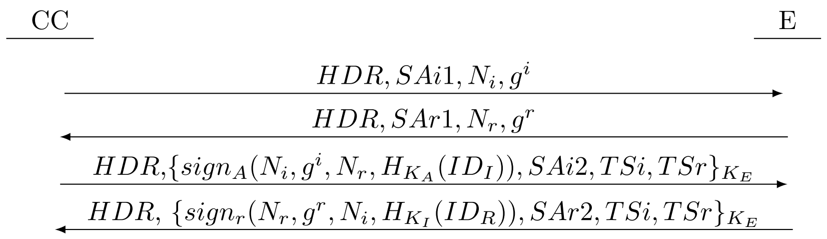
CC
E
$HDR, SAi1, N_i, g^i$
$HDR, SAr1, N_r, g^r$
$HDR, \{sign_A(N_i, g^i, N_r, H_{K_A}(ID_I)), SAi2, TSi, TSr\}_{K_E}$
$HDR, \{sign_r(N_r, g^r, N_i, H_{K_I}(ID_R)), SAr2, TSi, TSr\}_{K_E}$

**Figure 2.3:** IKEv2 as standardized in [KHNE14]. SAs contains suggestions of cryptographic primitives. HDR contains control information, e.g. SPI and flags.

A non-interactive public key distribution protocol was first suggested by Maurer and Yacobi [MY96]. If a peer's identity is known by an entity in advance, it can non-interactively use Diffie-Hellman (DH) key exchange to construct a shared key with this peer since the identity is its public key. Reference [SD03] argues that this advantage might not be fully exploited in IKE as the parties would have to exchange fresh nonces and other negotiated data anyway. Instead, the entities could use another key exchange protocol without the nonce exchange to take advantage of the non-interactive key distribution. The resulting shared secret key could further be used in pre-shared key mode of IKE [SD03]. IKE can be performed with, and without confidentiality protection of identities against passive adversaries. If identity protection is required, the parties exchange DH-exponentials initially and derive a shared key to protect the confidentiality of further exchanged information. This version requires one additional message because sending the identity- and authentication information must be delayed. If IP-addresses are used for identities, these must be known to the peer beforehand if such protection is desirable, as sending them in the first exchange would expose one's identity. Figure 2.4 shows how an IPSec SA could be set up between a CC and an E-node at the edge of a PCS.

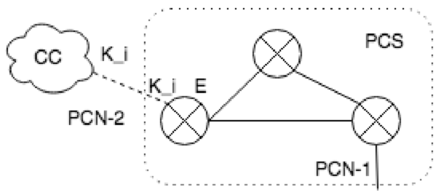

**Figure 2.4:** Figure showing an IPSec [KS05] security association set up between a CC and an edge E-node using IKE or KINK.

**Kerberos and ticket-based authentication** Kerberos is an authentication system based on symmetric key cryptography [MNSS88]. In Kerberos, a client authenticates once to an Authentication Server (AS), and receives a Ticket Granting Ticket (TGT) with a given expiration time. A TGT is a statement from the AS stating that the client has authenticated. The TGT is later used by the client to authenticate to a Ticket Granting Service (TGS) where the client requests a Service Ticket (ST) for a particular service on the network. After verifying that the TGT presented by the client is valid and not expired, TGS issues a ST and session keys to the client. The client then passes ST to an application server with which it wants to communicate securely. The three steps are illustrated in Figure 2.5. Kerberos is based on the Trusted Third Party (TTP) model for distributing keying material, and is in its basic form purely built on symmetric key cryptography. Several extension has been suggested and standardized to extend its scope, such as the use of public key cryptography for initial authentication [ZT06]. The use of Kerberos [NHYR05] for setting up security associations for IPSec is described in RFC 4430 KINK.

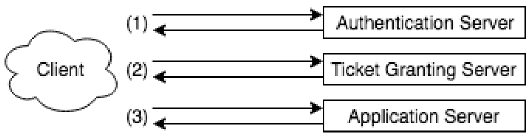

**Figure 2.5:** Kerberos as standardizied in [MNSS88]. (1) is called the AS-exchange; (2) the TGS-exchange; and (3) the AP-exchange

The ticket-based architecture is identified as a solution to the *efficiency*-requirement. Doing authentication based on attributes is considered advantageous with respect to the *scalability*-requirement. With the introduction of KINK as a way of setting up security association for IPSec, a possible solution to the *interoperability*-requirement was found. However, in Kerberos [NHYR05], there is not mechanism for a client to push service attributes to the network and negotiate. Hence, the *security*-requirement is not sufficiently fulfilled.

# Chapter 3 Theory

The following chapter is intended to introduce concepts, and theoretical background used in the thesis. There are three main parts to this chapter. First, theory related to authentication and to the methodology used in chapter 5 is covered. Second, background on variants of the Kerberos protocol is treated. Third, a high level introduction to IDPKC is given.

## 3.1 Formalizing Authentication

Cryptographic protocols are inherently difficult to get right. Many examples exist of protocols that have been deployed or standardized, just to later be shown insecure. The usage of formal methods to evaluate protocols have just started to receive wide spread acknowledgement. Two examples being the analysis of ISO/IEC 9798 in [BCM13]. Another example is the analysis of ISO/IEC 11770 in [CH14]. Contrary to the openness of the standardization process of cryptographic algorithms, cryptographic protocols have often been standardized without thorough study from the community [MMOB10]. Several tools for formal verification have emerged in recent years, and the process of correctly using these has evolved. In [MMOB10] Matsuo et.al. create a framework for security assessment of cryptographic protocols.

Reference [MMOB10] specifies that in order to get a protocol certified, the designer is required to create a document consisting of adversarial model, protocol specification, and a specification of security properties the protocol should satisfy. Finally, it must present some evidence that the protocol satisfies the promised properties in the given adversarial model. To reach the highest level of protocol assurance, all the above mentioned characteristics must be formally defined, and a verification tool with unbounded number of sessions used for evidence of correctness. Before formalizing security goals, a much used mechanism for identification called challenge-response will first be defined.

### 3.1.1 Challenge-response identification

The idea of using challenge-response of identification has been around for a long time. In [Dif88] Diffie claim that it dates back to 1952 when Horst Feistel at the Air Force Cambridge Research Center used it for identification of aircrafts to be friend or foe. Instead of using static challenge-response pairs, a radar challenged aircrafts with random numbers, and would determine their identity based on a correctly received encrypted response.

When discussing protocols, an *initiator* $I$ is the entity sending the first message of a protocol. A *responder* $R$, responds to an initiators request. In a later run of the protocol, the roles may change and the entity that initially took the responder role may take the initiator role. An identification scheme is an algorithm that allows an entity $R$ to prove to an entity $I$ that it is in fact $R$ [HPSS08]. Typically, $R$ wants to prove possession of a secret key to $I$ without revealing the key itself. A common way of doing this is through a challenge and response cycle [HPSS08]. Assume entities $I$ and $R$ share a symmetric key. Entity $I$ sends a fresh random challenge to $R$, on which $R$ applies some cryptographic operation using its key and sends it back to $I$. Using the same key, $R$ verifies that an operation was done on the challenge with the key known only by $I$ and $R$. The setup of challenge-response identification is illustrated in Figure 3.1. It is folklore that any challenge-response identification scheme built on public key cryptography can be converted into a signature scheme. In that case, as the challenge from $I$, $R$ uses a hash of a message. For the response, $R$'s signature over the message is used. Using a hash function over the message as a challenge assures $I$ that $R$ did not have any prior knowledge of the challenge as hash functions are one-way functions.

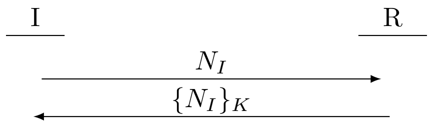

**Figure 3.1:** Challenge-response protocol based on fresh numbers called nonces as described in [Dif88]. Each nonce is randomly chosen, and used exactly once.

By sending a new challenge each time, an attacker is unable to *replay* an old response, and use it to fool the challenger. Since only a peer with the correct key is able to encrypt properly, identification of $R$ is achieved by proving possession of the encryption key $K$. One of the first works on identification schemes for computer networks is due to Needham and Schroeder in their work from 1978 [NS78]. Their public key unilateral identification scheme is illustrated in Figure 3.2

In their version of a public key based challenge-response mechanism, they sug-

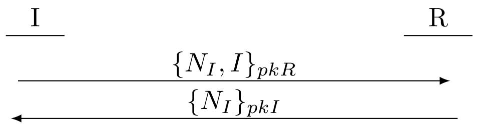

**Figure 3.2:** Unilateral Needham-Schroeder public key challenge- response [NS78]. The protocol was later shown insecure by Lowe [Low96]

gested including the sender's identity under the encryption. Since anyone could obtain $R$'s public key, including the identity $I$ gives $R$ assurance that it is in fact $I$ that wants to communicate. However, later Lowe [Low96] showed that the protocol was vulnerable to a man-in-the-middle attack. By initiating a session with $I$, and later impersonating $I$ to $R$, an attacker was able to perceive $R$ to believing that it was in fact communicating with $I$. The fixed version incorporates an advice that was later stated by Abadi and Needham in [AN96]. They advised that each protocol message should clearly state what it means, and that its interpretation should only depend on its content. Based on that, the identity of the responder $R$ was included in the response.

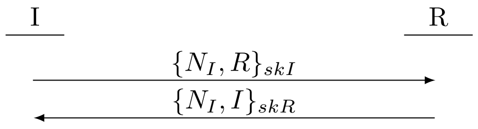

**Figure 3.3:** Needham-Schroeder-Lowe public key two-party unilateral challenge-response based on nonces [Low96].

A second version of the public key Needham-Schroeder-Lowe (NSL) protocol was also suggested using digital signatures. This version, illustrated in Figure 4.4, is the one used in chapter 4. Note that the entities here includes the identity of its peer under the signature instead of its own as in Figure 3.2.

### 3.1.2 Security goals

In [BM13] Boyd et.al. define the terms *user-oriented goals* and *key-oriented goals* to describe goals related to authentication and key establishment respectively. The focus here will be on user-oriented goals. That is, goals related to entity authentication rather than properties of the established key.

It is not straight forward what is meant by authentication. A common definition from [Low97] called *weak aliveness* is now stated as one example. "A protocol provides weak aliveness to an initiator $I$ of a responder $R$ if $I$ observes $R$'s identity, and only passive eavesdropping adversaries are assumed". Entity $I$ can then be sure that $R$ was at the network at some point in time. Another useful definition would be to give $I$ assurance that $R$ is in fact performing the responder role of the communication session. These are both definitions of entity authentication. It is clear that the assurance one entity receives from its believed peer varies greatly in these definitions. Cremers define in [Cre06] a degree of authentication called synchronization which is proved stronger than Lowe's definition of injective agreement. The difference being that Lowe's definition does not concern itself with the order of a *send*- and *receive*-event. However, synchronization explicitly define that a send must happen before a receive. This opens up the possibility of modelling *preplay attacks* [Cre06]. A preplay means that a message was injected into the network before it was formally created. Sometimes it may be difficult to objectively determine if an attacker has been active in a session. One way is to see if a given protocol would finish differently if an adversary was present. Another way, which will be discussed shortly, is how each participant view the scheme carried by the protocol. As neither participant has information on the global state of the scheme, each participant only knows what is provided through the protocol or generated locally. The following definitions of authentication is taken from [Low97] and will be used in chapter 5.

The goal below says nothing about if $R$ knows that it is running the protocol with $I$ at all. Neither does it say anything about when $R$ has run the protocol.

**Definition 3.1. Recent Aliveness:** A protocol guarantees to an initiator $I$ aliveness of another agent $R$ if, whenever $I$ (initiator) completes a run of the protocol, apparently with responder $R$, then $R$ has previously been running the protocol.

It does not say anything about *when* $R$ was running the protocol, nor that $R$ was responder in the same protocol instant.

**Definition 3.2. Weak Agreement:** A protocol guarantees to an initiator $I$ weak agreement with another agent $R$ if, whenever $I$ (initiator) completes a run of the protocol, apparently with responder $R$, then $R$ has previously been running the protocol, apparently with $I$.

In non-injective agreement $R$ does not necessarily agree with $I$ on the number of sessions they are running.

**Definition 3.3. Non-injective Agreement:** A protocol guarantees to an initiator $I$ non-injective agreement with a responder $R$ on a set of data items $ds$ if, whenever $I$ (initiator) completes a run of the protocol, apparently with responder $R$, then $R$ has previously been running the protocol, apparently with $I$, and $R$ was acting as responder in this run, and the two agents agreed on the data values corresponding to all the variables in $ds$.

*Injective* agreement means that there is a one-to-one relationship between the sessions run by $I$ and those run by $R$.

**Definition 3.4. Injective Agreement:** A protocol guarantees to an initiator $I$ agreement with a responder $R$ on a set of data items *ds* if, whenever $I$ (initiator) completes a run of the protocol, apparently with responder $R$, then $R$ has previously been running the protocol, apparently with $I$, and $R$ was acting as responder in his run, and the two agents agreed on the data values corresponding to all the variables in *ds*, and each such run of $I$ corresponds to a unique run of $R$.

### 3.1.3 Adversary model

An adversary model says something about the capabilities and intent of an attacker. The Dolev-Yao (DY) adversary model [DY83] assumes an active eavesdropper, who after having gained access to the communication channel has access to all messages sent on the channel. The adversary can alter, inject, delay, or drop packets, and can also be a legitimate party and initiate connections with others. Furthermore, cryptographic primitives are assumed to be perfect. That is, a party not in possession of the secret key is not able to change or forge a digital signature, or decrypt encrypted data. Compromising adversaries are however not assumed. Attackers are therefore modelled without being in possession of long term secrets.

### 3.1.4 Tools for formal analysis of protocols

Scyther-proof and the underlying scyther were chosen in this thesis due to their state-of-the-art performance and ease of use. To achieve increased trustworthiness of results, scyther-proof can check its output proof script in Isabelle [Mei13].

**Scyther** Scyther [Cre08] is a tool developed by Cas Cremers for unbounded verification and falsification of security protocols. The tool provides state-of-the-art performance [Cre08], and guaranteed termination. Additionally, it can also provide proof that no attack exists within a given bound. In scyther, protocols are modelled using the Security Protocol Description Language (SPDL) language, based on the semantics in [Cre06]. Security properties in scyther are modelled as so called *claim events*. A claim is based on the local view of a participant on a set of values. Thus, based on locally generated values, and messages received, an entity claims some security goal. The purpose of the protocol is for an agent to be sure that some global property hold, based on just local view[Cre06]. Scyther is available for download at tool website[1], or on the developers GitHub account[2].

[1] https://www.cs.ox.ac.uk/people/cas.cremers/scyther/
[2] https://github.com/cascremers/scyther

**Scyther-proof** Scyther-proof was developed in an attempt to increase trustworthiness in protocol verification tools. In [Mei13] Meier outlines a generic approach to construct algorithms for theorem proving from security protocol verification algorithms. In that respect, scyther-proof is a demonstration of the generic approach applied to scyther. Scyther-proof generates human readable proof, which can further be analyzed using Isabelle/Higher-Order Logic (HOL) theorem prover. Being able to independently analyze the proofs using a theorem prover like Isabelle/HOL increases credibility of results from scyther-proof. Proofs are generated in scyther-proof by first assuming that an attack exists. In order to perform some attack, some information must be obtained by the adversary. By studying each case where an attacker might have obtained this information, and repeating this process recursively, a theorem can be proved if all instances of an attack leads on a contradiction. The theory underlying scyther-proof is taken from reference [Mei13]. The security protocol model underlying scyther-proof consists of a protocol specification language, an operational semantics of how the protocol executes in the presence of an active adversary, and security properties [Mei13]. Security protocols with scyther-proof are modelled as a set of roles. Each of these roles execute a sequence of role steps. A role step will consist of either a send or a receive event. When receiving or sending a message from the network, an entity will match the given message to a pre-defined message pattern to identify its structure.

## 3.2 Kerberos authentication

Four-pass Kerberos and public key Kerberos are covered in this section. Kerberos builds on the work by Needham and Schroeder on TTP authentication [NS78]. Specifically, authentication depends upon the two entities, $A$,$B$, and a Key Distribution Center (KDC) being the only ones who know the session key that is being used.

### 3.2.1 Four-pass Kerberos

The four-pass Kerberos is here described as in [Sti05]. Entity $A$ takes the role of client in Figure 3.4, $B$ that of application server, and Trusted Authority (TA) the role of AS. The KDC incorporates the functionality of both the authentication server and ticket granting server in standard Kerberos. When receiving $A$'s request, the $TA$ generates a fresh session key $K$. The $TA$ also specifies for how long the session key should be valid, i.e. the lifetime $L$. A random challenge $r_A$ sent by $A$ is included in the response from $TA$ to $A$. The random number is included to prevent an attacker from replaying and old response. Since the entire response from $TA$ is encrypted with a key shared between $A$ and $TA$ only, $A$ is able to authenticate $TA$. Note that $TA$ does not authentication $A$ as all values are received unencrypted. When approaching the server $B$, $A$ relays the ticket $t_B$ and sends an authenticator to $B$. The authenticator $y_2$ includes $A$'s identity, and the current time encrypted with the session key. $A$ proves knowledge of the session key through the authenticator. By proving that it knows the session key, $A$ also proves that it was able to decrypt $y_1$, and hence is authenticated by $B$.

When $B$ decrypts $t_B$, it uses the session key to decrypt the authenticator from $A$. If only $A$ and $TA$ know their shared key, and one unidentified entity proves to $B$ that it knew this key and used it to retrieve the session key, $B$ can conclude that his session key is meant for communication with $A$. Finally, $B$ proves to $A$ that it was able to decrypt $y_1$ and hence knows the session key. It does so by incrementing the timestamp and sending it encrypted with the session key back to $A$.

1. $A$ chooses a random number, $r_A$. $A$ sends $ID_A$, $ID_B$ and $r_A$ to the TA.
2. $TA$ chooses a random session key $K$, a lifetime, $L$ and computes a ticket to $B$, $t_B = e_{K_B}(K\|ID_A\|L)$, and $y_1 = e_{K_A}(r_A\|ID_B\|K\|L)$. $TA$ sends $t_B$ and $y_1$ to $A$.
3. $A$ decrypts $y_1$ using its key $K_A$, obtaining $K$. Then $A$ determines the current time, $time$, and computes $y_2 = e_K(ID_A\|time)$.
4. B decrypts $t_B$ using its key $K_B$, obtaining $K$. It also decrypts $y_2$ using the key $K$, obtaining $time$. Then $B$ computes:
$y_3 = e_K(time + 1)$. Finally, $B$ sends $y_3$ to $A$

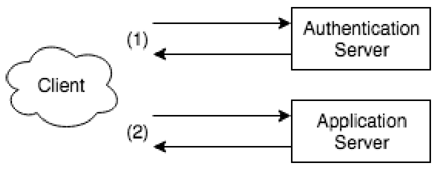

**Figure 3.4:** Four-pass Kerberos [Sti05] where a client immediately requests KDC for a ST instead of a TGT. (1) Is called the AS-exchange; (2) the AP-exchange

### 3.2.2 Public key Kerberos

Using public key cryptography for entity authentication reduces the number of entities a client needs to share a key with at deployment. With Public Key Cryptography for Initial Authentication (PKINIT) as illustrated in Figure 3.5, only the AS-exchange of Kerberos is changed, that is the first two messages. In Figure 3.5 $KDC$ symbolizes the identity of a KDC, $N_C$ a nonce, and $T_C$,$T_1$, $T_N$ are timestamps representing the lifetime, start and end-time respectively. In the second message, $K$ is a new session key, and $MAC_K(*)$ is a Message Authentication Code (MAC) keyed with $K$ over the entire initial request. $S$ represent the application server for which $C$ has been issued a ticket.

$$\begin{tikzcd} \text{C} \arrow[r, shift left, "(*)sign_C\{KDC, T_C, N_C\}"] & \text{KDC} \arrow[l, shift left, "\{sign_{KDC}\{K, MAC_K(*)\}\}_{pubC}, C, ST, \{K_{CS}, N_C, T_1, T_N, S\}_K"] \end{tikzcd}$$

**Figure 3.5:** A simplified version of PKINIT [ZT06] is illustrated. Public key cryptography based on certificates are used to exchange a session key.

## 3.3 Identity-based cryptography

Identity-based cryptography is a paradigm within public key cryptography where an entity's public key is derived from its identity. By having the mapping between identity and public be trivial, no external mechanism for authenticity is required. In the article introducing the idea of identity-based cryptography Shamir points out that this concept is ideal for closed user groups [Sha85]. A high level description of its setup and usage is given here.

In [JN09] Kiltz et.al. discuss characteristics of IDPKC. Because the public key is a globally known value, it should not be possible to compute the private key from the public key given only public system parameters. They further argue that since this is an identity-based system, it should not be dependent on some user-specific value other than the identity. Unlike RSA cryptosystem [RSA78] where each user chooses two secret prime numbers and use them as part of the cryptosystem. If a similar approach was adopted in IDPKC, it would be difficult to find a mapping between the public identity and a unique decryption with that key. As a consequence, the computation of private keys must be a function of a global trapdoor information common to many users. The entity in possession of the global trapdoor information in IDPKC is called KGC. The inherent property of IDPKC requiring a central KGC enables key escrow. That is, a KGC could in theory forge signatures or decrypt traffic for all entities with a key registered with that KGC.

IDPKC the names Setup and KeyGen are used for the required algorithms specific to the identity-based paradigm. The setup algorithm run by KGC produces two keys, Master Secret Key (MSK) and Master Public Key (MPK). MPK is published and made available to all users in a security domain, while MSK is kept secret. In the KeyGen algorithm individual users' private keys are generated. After deciding on an identity to use, a user registers this key with the KGC. After verifying the user's claim on its identity, the KGC combines the identity with MSK to create the corresponding private key, called User Secret Key (USK). The user then receives the private key from KGC and stores it at a secure location. Other users with knowledge of a peer's identity can verify a signature from the peer, or encrypt information to the peer using its identity as the public key. There is a significant difference between

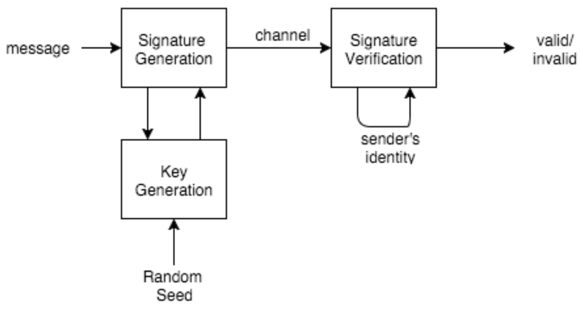

**Figure 3.6:** A high level description of digital signatures using IDPKC as described in [Sha85].

algorithms where the owner of the private key initiates (e.g. signatures such as in Figure 3.6), and when the owner of the public key initiates (e.g. encryption as in Figure 3.7). In the former case, such as when signing, all information required by the receiver can be embedded in the message. This is however not the case with the process of encryption, where the initiator only has the identity in order to encrypt. Auxiliary information would then have to be extracted from the receiver's identity.

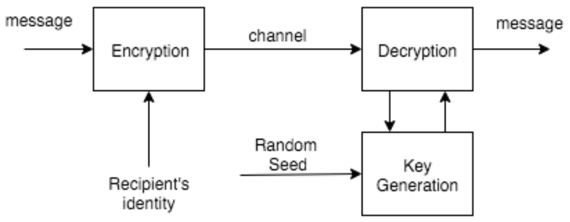

**Figure 3.7:** A high level description of encryption and decryption using IDPKC as described in [Sha85].

Identity-based Signature (IBS) will be studied in more detail in the following section. The use of Identity-based Encryption (IBE) in PCN will however not be studied further in this thesis in order to limit the scope. IBE requires a special construction called a *pairing* on elliptic curves to be realized. A definition of this term will be give shortly, since its use in IBS will be discussed in chapter 6.

### 3.3.1 Identity-based signature algorithms

The following definition of IBSs is borrowed from [GG09]. The Setup algorithm is referred to as *parameter-generation algorithm.* KeyGen is referred to as *key-extraction algorithm.* Then, an IBS scheme is a quadruple $(\mathcal{G}, \mathcal{E}, \mathcal{S}, \mathcal{V})$ of probabilistic polynomial-time algorithms, where

- $\mathcal{G}$ is the *parameter-generation algorithm.* It takes as input the security parameter $1^\eta$ and outputs the system public parameters **MPK**, a master secret key **MSK**.
- $\mathcal{E}$ is the *key-extraction algorithm* that takes as input parameters **MPK**, a master key **MSK** and an identity **id** and outputs a private key $usk_{id}$ corresponding to the user with this identity.
- $\mathcal{S}$ is a *signing algorithm* that takes as input parameters **MPK**, a private key $usk_{id}$ and a message $m$ and outputs a signature $\sigma$.
- $\mathcal{V}$ is a deterministic *signature verification algorithm* that takes as input parameters **MPK**, a signature $\sigma$, a message $m$ and an identity **id** and outputs whether or not $\sigma$ is a valid signature of $m$ relative to (**MPK**,**id**).

Several IBS-schemes can be realized using number theoretic primitives. Other require a special construction called a *pairing* on elliptic curves.

### 3.3.2 Identity-based signatures from number theory

A way of transforming a class of identification schemes to IBS-schemes was found and proved independently by Bellare et.al. [BNN09] and Kurosawa and Heng [KH04] in 2004. Instances of these identification schemes exist for hard problems based on both factoring, pairings, as well as the RSA. The Fiat-Shamir (FS) transformation [FS86] defines a transformation of a standard identification scheme to a standard signature scheme. Reference [BNN09] builds on this further by creating a transformation of a standard identification scheme to an identity-based identification scheme. They further define a transformation that takes convertible standard signature schemes, and transforms them to identity-based signature schemes. The result is a corollary that states that a special class of convertible standard identification schemes can be transformed to identity-based signature schemes through the two transformations.

One identification scheme with special properties which has a corresponding signature scheme is Schnorr, which will be explained in the next section.

### 3.3.3 Schnorr signature scheme

The description of Schnorr signature scheme is copied from [Sti05]. The scheme functions as a basis for two identity-based schemes described in chapter 6.

Let $p$ be a prime such that the discrete logarithm problem in $\mathbb{Z}_p^*$ is intractable, and let $q$ be a prime that divides $p-1$. Let $\alpha \in \mathbb{Z}_p^*$ be a q'th root of 1 modulo $p$. Let $\mathcal{P} = \{0,1\}^*$, $\mathcal{A} = \mathbb{Z}_q \times \mathbb{Z}_q$, and define:

$\mathcal{K} = \{(p, q, \alpha, a, \beta) : \beta \equiv \alpha^a (modp)\}$,

where $0 \leq a \leq q-1$. The values $p, q, \alpha$, and $\beta$ are public key, and $a$ is the private key. Finally, let $h : \{0,1\}^* \to \mathbb{Z}_q$ be a secure hash function.

For $K = (p, q, \alpha, a, \beta)$, and for a (secret) random number $k$, $1 \leq k \leq q-1$, define: $sig_K(x, k) = (\gamma, \delta)$ where $\gamma = h(x \| \alpha^k modp)$ and $\delta = k + a\gamma \bmod p$.

For $x \in \{0,1\}^*$ and $\gamma, \delta \in \mathbb{Z}_q$, verification is done by performing the following computations: $ver_k(x, (\gamma, \delta)) = \text{true} \Leftrightarrow h(x \| \alpha^\delta \beta^{-\gamma} modp) = \gamma$

### 3.3.4 Identity-based signatures from pairings

A pairing is a function that maps points of an elliptic curve into the multiplicative group of a finite field. Or more generally, a pairing $\hat{e}$ is a map from a group $\mathbb{G}$ to another group $\mathbb{G}_T$ The following explanation of bilinear pairing is from [JN09]:

$$\hat{e}\colon\ \mathbb{G} \times \mathbb{G} \mapsto \mathbb{G}_T \tag{3.1}$$

Such that for all elements $P$, $Q$ in $\mathbb{G}$, and all integers a and b, we have $\hat{e}(aP, bQ) = \hat{e}(P, Q)^{ab}$. Typically $P, Q$ are points on an elliptic curve $E$, defined over a finite field $F_q$. $P$ and $Q$ are assumed to be points of order r on $E$. $Q$ may be defined over an extension field of $F_q$ The map is also specified to be non-degenerate, which means that all elements should not be mapped to the identity in $\mathbb{G}_T$. Also required is a hash function that maps any identity $id$ to a point $\mathrm{P}_{id}$ in $\mathbb{G}$ in a collision resistant manner. The KGC creates a secret $s$, and publishes a triple $(P, Q, \mathcal{Q})$, where Q, and $\mathcal{Q}$ are related by $Qs = \mathcal{Q}$. When a user whose identity maps to $P_{id}$ asks KGC for its private key, it receives back the point $\mathcal{P}_{id} = P_{id} \cdot s$

**Definition 3.5. Embedding Degree** (Directly from [KM05]) Given an elliptic curve $E : y^2 = x^3 + ax + b$ defined over the finite field $\mathbb{F}_q$, and a basepoint $P$ with prime order $n$ dividing $E(\mathbb{F}_q)$. Assume further that $n$ does not divide $q$. Let $k$ be the multiplicative order of $q$ modulo $n$. That is, the smallest positive number $k$ such that $n \mid q^k - 1$. The number $k$ is called the *embedding degree*.

Even though all elliptic curves have pairings, not all are useful in cryptographic applications [MPP+15]. Definition 3.5 introduces the concept of a curves embedding degree. In order for an elliptic curve to have pairings that can be implemented in practice, the curve needs to have a small embedding degree. In addition to finding a proper curve, the pairing itself need to be chosen. The most common pairings today are the Weil and Tate pairings [KM05].

**Definition 3.6. Bilinear Diffie-Hellman Problem** Let $\mathbb{G}_1$ and $\mathbb{G}_2$ be two cyclic groups of prime order $m$, and let $P$ be a generator of $\mathbb{G}_1$. Let $\hat{e}$: $\mathbb{G} \times \mathbb{G} \mapsto \mathbb{G}_T$ be a bilinear map. Given $P$, $aP$, $bP$, $cP$ for some $a, b, c \in \mathbb{Z}_m^*$, compute $v \in \mathbb{G}_2$ such that $v = e(P, P)^{abc}$

**Definition 3.7. Discrete Logarithm Problem**
Assuming $(G, \cdot)$ is a multiplicative group and $\alpha \in G$ has order n. The discrete logarithm problem is: Given $\beta \in \langle\alpha\rangle$, find the unique exponent $a$, $0 \leq a \leq n - 1$, such that $\alpha^a = \beta$

Definitions 3.6 and 3.7 will not be commented here, but they are used in a discussion regarding efficiency of pairings in chapter 6. The next chapter introduces two new protocols for PCN with application of IDPKC.

Chapter 4

# Proposal of two new authentication protocols for PCN

The following chapter propose two new extensions to Kerberos for its use in PCN. One protocol is based on four-pass symmetric key Kerberos, called *A*, and the other on four-pass PKINIT Kerberos, called *B*. Separate authentication protocols are run between entities, and carried inside the Kerberos messages, to achieve stronger security guarantees. The authentication protocols used within *A* and *B*, are called *a* and *b* respectively. Protocols *a* and *b* are authenticated using identity-based signatures and use timestamps and nonces respectively to assure freshness. Focus is kept on the cryptographic core of the proposed protocols.

In Kerberos notation, a CC will act as a client, the KDCs as authentication servers, and E-nodes as application servers. The KDC in a CC's home PCS is referred to as HKDC, and VKDC in the visiting PCS. HKDC and VKDC are assumed to support both public key cryptography using certificates and IDPKC. From this point on, the Kerberos terminology and the PCN-equivalent will be used interchangeably. In Figure 4.1 a CC is assumed to have moved geographically, and reconnected to PCore through an E-node in a visiting PCS. Similarly to Kerberos, a CC first approaches VKDC when connecting to a new PCS. The E-node at the edge in Figure 4.1 is assumed to forward authentication traffic to the VKDC. Traffic belonging to other services is blocked by the edge E-node. After finishing communicating with VKDC, a CC approaches the E-node for the last part of connection setup.

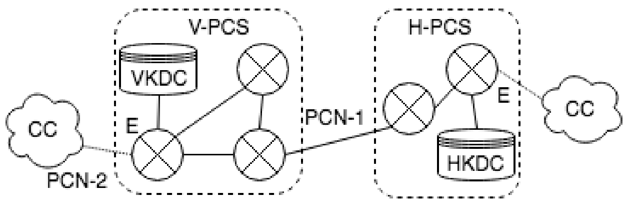

**Figure 4.1:** A CC connects to PCore through an E-node in a visiting PCS. VKDC is responsible for initial authentication, before a CC approaches an E-node.

## 4.1 Attribute-based authentication and protocols *a* and *b*

The following section explains how data carried by protocols *a* and *b* are separately authenticated in using identity-based signatures.

The challenge of distributing service policy is solved within protocols *a* and *b*. First to VKDC from HKDC, then from VKDC to the correct E-node. Similarly to the ISs proposed by Fongen [Fon10], it is assumed that CCs carry documents stating their service rights issued by HKDC. These documents will be referred to as *service statements*. A service statement will contain attributes relevant for a CC, as well as a validity time, identity of issuing KDC, and which CC it is issued to. A service statement is authenticated using identity-based signatures. In order to involve CCs in the process of requesting a specific service level, CCs encapsulate service statements issued by KDCs in ones created by itself. The outer service statement represents the CCs request for service, while the inner represent allowed service authorized by a KDC. The idea is clarified by Figure 4.4. When approaching a visiting PCS, a CC encapsulates the service statement issued by HKDC inside one created by itself containing requested service. Upon receiving this statement from a CC, VKDC verifies the signatures and checks if the CC requests more than it is authorized to receive. It further issues a new service statement for the CC, on which services it can use in V-PCS. The process is repeated when the CC connects to an E-node. CC encapsulates the service statement received from VKDC into one created by itself on requested service. After verifying the signatures and checking that the CC did not request more than decided by VKDC, the E-node returns a service statement on which services the CC can expect. A relevant question is how CC and E-node are able to verify the signatures created by one another, when little pre-established relationship is assumed. The solution is found in cryptography, rather than in authentication protocol theory and is introduced in the next paragraph.

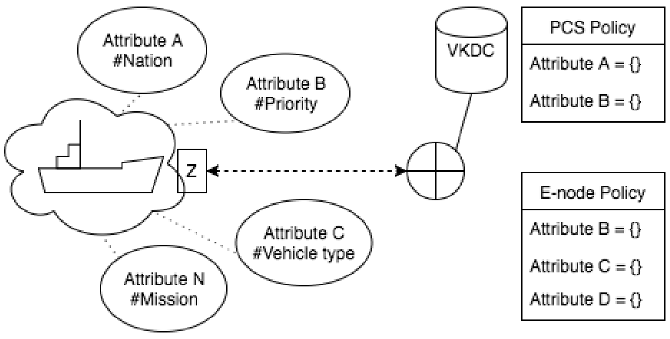

**Figure 4.2:** A figure showing how attributes of a CC can be used for authentication. The attributes are linked to service policies in both VKDC and E-node

IDPKC decouples the public key and the private key. Any string, such as an identity, can be used as a public key. Instead of using CCs' unique identities as public keys, sets of service attributes can constitute public keys. Since service attributes are sent as part of service statements, E-nodes are able to reconstruct public keys, and verify signatures from CCs. E-nodes would then not verify CCs' identities, but rather verify that they are authorized for a given set of service attributes. Figure 4.2 illustrates how different characteristics of a CC may be used for authentication. A ship with PCN capabilities dock at a harbor of a nation managing a PCS. Certain attributes are associated with the ship, such as nation, priority, type of ship, or mission name. When a CC connects to the PCS, VKDC determines PCS-global policy and sends it back to the CC. Afterwards, when connecting to an E-node, the service decision made by VKDC is tempered based on local state at the E-node.

Two different schemes are identified. One approach is to associate a private key directly with a pre-defined service level. Another is to let multiple attributes constitute a service level, and a CC could have one private key for each service attribute. One scheme for each of these two alternatives will be discussed in chapter 6. In Figure 4.3, (1) is a PKI where policy is stored in routers in advance. Row (2) illustrates IDPKC where a policy is linked to a user's identity. In (3), a private key is linked directly to a given service level. With (4), multiple attributes constitute a service level, and each attribute is coupled with a private key. E-nodes are assumed

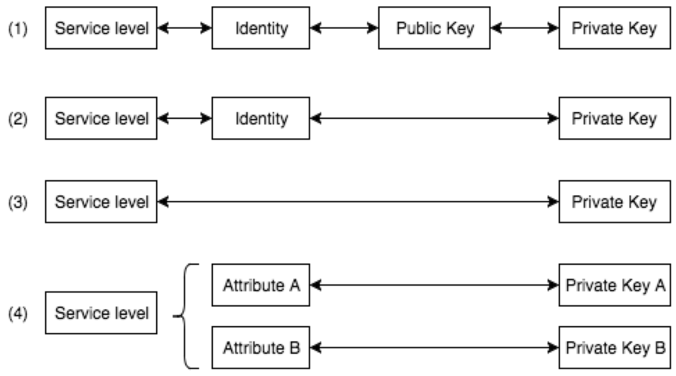

**Figure 4.3:** Four way of coupling private keys and service classes are shown. (3) directly links a key with a service level. Alternative (4) may allow more flexibility

familiar with the identity of the KDC in its own PCS. Furthermore, KDCs from different PCSs are assumed to know each other's identities in advance. The proposed solution of using service attributes for identities is here assumed mainly between a CC and an E-node, as they are least likely to share each other's identities in advance. Protocols *a* and *b* are now introduced. Both follow the high level steps explained above when a CC connects to a visiting PCS.

**Protocol *a*:** A CC has previously received from HKDC a service statement containing service attributes associated with it, further referred to as *Auth1*. Together with *Auth1*, information on issuing KDC, identity of $CC$, and lifetime $L0$ it is signed by HKDC, $sign_{HKDC}(HKDC, CC, Auth1, L0)$. When connecting to a visiting PCS, a CC presents the service statement to VKDC, embedded within a new service statement. In Figure 4.4, *Auth2* represents the CC's request for service, and $L1$ is the current time. After receiving this element, the VKDC verifies the signature, checks that $Auth2$ is a subset of $Auth1$, and that it can provide the services requested in $Auth2$. VDKC also checks if $L1$ is an old timestamp, and hence if the message is a replay. VKDC's decision on allowed service is sent back to the $CC$ in a data element called *Auth3*. In Figure 4.4 VKDC includes both a start time $T1$, and an expiration time $TN$ in the new service statement for the visiting PCS.

$$\begin{tikzcd} \underline{\text{CC}} \arrow[rr, shift left, "sign_{CC}(sign_{HKDC}(HKDC, CC, Auth1, L0), Auth2, CC, L1)"] & & \underline{\text{VKDC}} \arrow[ll, shift left, "sign_{VKDC}(VKDC, CC, Auth3, T1, TN)"] \end{tikzcd}$$

**Figure 4.4:** Negotiating version for access rights where the client receives assurance on which services the visiting PCS supports.

A similar exchange is performed when a CC connects to a specific E-node. The element $sign_{CC}(sign_{VKDC}(VKDC, CC, Auth3, T1, TN), Auth4, CC, L2)$ is then sent from the CC to an E-node. The element *Auth4* represents the $CC$'s request for service. A final decision taken by the E-node is echoed back within *Auth5*, as $sign_E(E, CC, Auth5, T2, TM)$. Elements $T2$ and $TM$ represent start time and expiration time respectively. In protocol $a$ as described here, the goal of the E-node is to receive information from both VKDC and CC on which services should be delivered. For CC, the goal is to actively request certain services and receive assurance from VKDC and E-node on which of these services will ultimately be delivered.

**Protocol *b*:** A CC first approaches VKDC with a request for service, and receives a service statement for the PCS. The CC uses this statement to request services from an E-node. However, instead of using timestamps under the signatures, a public key NSL challenge-response based on nonces is used. First between CC and VKDC, then between CC and E-node. The challenge-response is however only one-way, and gives assurance to CC that the response received from VKDC or E-node was not replayed. The service statements issued by HKDC and VKDC would still need to include timestamps to limit their validity. Also, when a CC sends requests to VDKC or an E-node, it would still need to include the current time to give them freshness assurance. These separate authentication exchanges are intended to give increased control to the CC over the decision made by the foreign network on expected service.

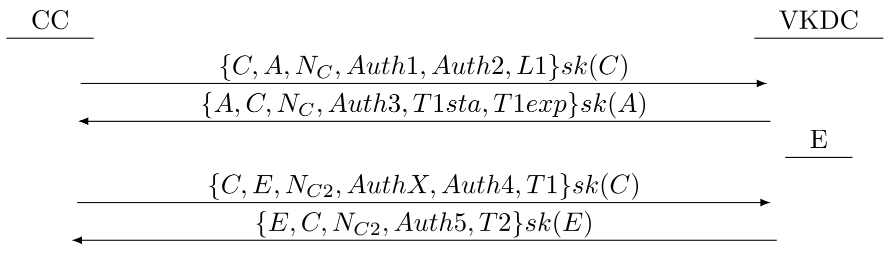
CC
VKDC
$\{C, A, N_C, Auth1, Auth2, L1\}sk(C)$
$\{A, C, N_C, Auth3, T1sta, T1exp\}sk(A)$
E
$\{C, E, N_{C2}, AuthX, Auth4, T1\}sk(C)$
$\{E, C, N_{C2}, Auth5, T2\}sk(E)$

**Figure 4.5:** Figure showing two public key NSL unilateral authentication protocol. Here {Auth$i$}sk(C) means a signature created by $C$ over Auth$i$

Protocol $b$ shown in Figure 4.5 is different from protocol $a$ in one more way. Instead of forwarding the entire service statement received by VKDC to an E-node, a CC here only use it for information. That is, instead of functioning as an authority, the VKDC gives advice to the CC on services provided by the PCS. As is illustrated in Figure 4.5, CC sends $AuthX$ to the E-node instead of $Auth3$ done in Figure 4.4. The element $AuthX$ may or may not be different from $Auth3$ received by VKDC, which only functions as information to aid a CC in requesting further services. Four-pass Kerberos with protocol $a$ inside is introduced first. Second, PKINIT Kerberos carrying protocol $b$ is treated.

## 4.2 Protocol A: Four-pass Kerberos with *a*

Protocol $A$ is based on four-pass Kerberos with symmetric key cryptography [CKM00]. In order to participate in authentication, a $CC$ push a service statement to the $VKDC$ in the first Kerberos message. The service statement is represented in Figure 4.6 with timestamps, identifiers, and service attributes Auth$i$ under the signatures. Message passes of protocol $a$ are included in four-pass Kerberos by simply including each messages pass within its Kerberos equivalent message number. In Figure 4.6 the service statement issued by HKDC to CC in the first message is exactly the same as the first message of Figure 4.4. No interaction between the timestamps used by HKDC, and those used by VKDC in the Kerberos protocol is assumed. However, it is assumed that the timestamps used by VKDC for service statements are the same as used within the tickets in Kerberos. In Figure 4.6 VKDC is abbreviated $A$, and CC as $C$ for readability. $L1$ is the current time, $N_A$ a nonce. Elements $T1sta$, and $T1exp$ are timestamps stating start time and expiration time for a Kerberos ticket. $T1$ and $T2$ are sequence numbers. Furthermore, $K_{CE}$ is the freshly generated server key intended for communication between $C$ and $E$. Next, protocol $b$ carried within PKINIT Kerberos is introduced

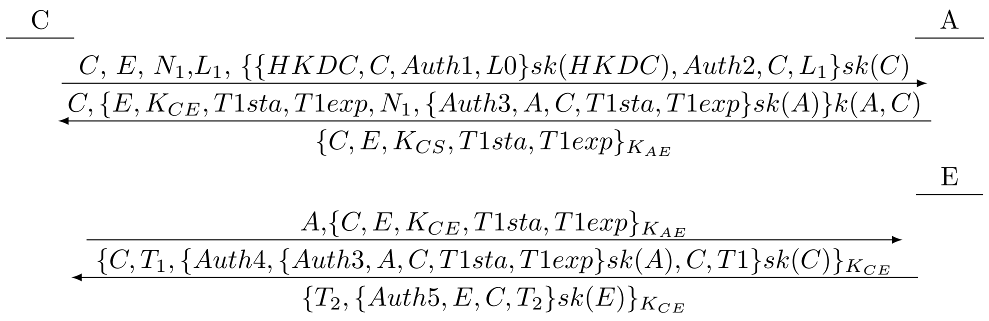
C
A
$C,\ E,\ N_1,L_1,\ \{\{HKDC,C,Auth1,L0\}sk(HKDC),Auth2,C,L_1\}sk(C)$
$C,\{E,K_{CE},T1sta,T1exp,N_1,\{Auth3,A,C,T1sta,T1exp\}sk(A)\}k(A,C)$
$\{C,E,K_{CS},T1sta,T1exp\}_{K_{AE}}$
E
$A,\{C,E,K_{CE},T1sta,T1exp\}_{K_{AE}}$
$\{C,T_1,\{Auth4,\{Auth3,A,C,T1sta,T1exp\}sk(A),C,T1\}sk(C)\}_{K_{CE}}$
$\{T_2,\{Auth5,E,C,T_2\}sk(E)\}_{K_{CE}}$

**Figure 4.6:** Figure showing protocol $A$, i.e. the composition of protocol $a$ and symmetric key Kerberos. The messages of protocol $a$ are carried within Kerberos.

## 4.3 Protocol B: Four-pass PKINIT Kerberos with *b*

In protocol $B$ a CC initially authenticates to VKDC using four-pass PKINIT Kerberos. PKINIT Kerberos is presented in Figure 4.7 without protocol $b$ for readability. Protocol $b$ would however be carried by PKINIT similarly to protocol $a$ in four-pass Kerberos described above. In Figure 4.7 CC is abbreviated as $C$, HKDC as $A$, and $E$ represents an E-node. It is assumed that certificates are used for authenticity in the PKINIT exchange between a CC and VKDC.

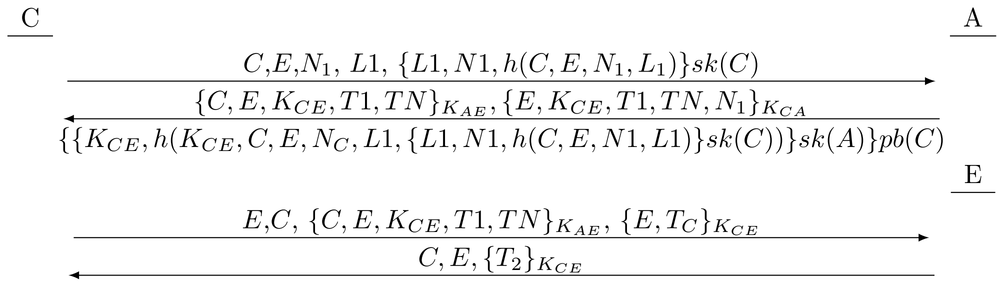
C
A
$C,E,N_1,\ L1,\ \{L1,N1,h(C,E,N_1,L_1)\}sk(C)$
$\{C,E,K_{CE},T1,TN\}_{K_{AE}},\{E,K_{CE},T1,TN,N_1\}_{K_{CA}}$
$\{\{K_{CE},h(K_{CE},C,E,N_C,L1,\{L1,N1,h(C,E,N1,L1)\}sk(C))\}sk(A)\}pb(C)$
E
$E,C,\ \{C,E,K_{CE},T1,TN\}_{K_{AE}},\ \{E,T_C\}_{K_{CE}}$
$C,E,\{T_2\}_{K_{CE}}$

**Figure 4.7:** Four-pass PKINIT Kerberos without protocol $b$. Here $h()$ is a hash function, and public key encryption of element $A$ by $C$ is represented as $\{A\}pb(C)$.

The advantage of protocols $A$ and $B$ as presented here is that an E-node only need to know the attributes of a CC in advance, and not globally unique identities. Since authentication of service attributes can be performed between all three entities, they can achieve greater agreement on service policy. This claim, as well as security properties of the protocols, will be formally verified in the next chapter.

# Chapter 5 Verification of protocols using formal methods

Following the standardized approach to achieve protocol assurance specified in chapter 3, the chapter is outlined as follows. The protocols *A* and *B* are first formally described using SPDL. Furthermore, security goals such as authentication and secrecy are defined. Finally, security goals and protocol models are input to the tool scyther-proof [Mei13] for formal verification. The DY adversary model is assumed, and is built into scyther-proof. The DY-model was defined in chapter 3.

## 5.1 Protocol Descriptions

The following section contains the models of the protocols from chapter 4 in SPDL. Table 5.1 defines the most common syntax when modelling protocols in SPDL.

| Cryptographic syntax of scyther-proof | |
|---|---|
| Identity of peer *I* | I |
| Fresh Nonce generated by peer *I* | Ni |
| Hash function | h() |
| Timestamp | T1 |
| MAC of message *M*, under key *k* | h(M, k(I,R)) |
| Digital Signature of peer *I* on message *M* | {M}sk(I) |
| Asymmetric private key of peer *I* | sk(I) |
| Asymmetric public key of peer *I* | pb(I) |
| Asymmetric encryption of peer *I* on message *M* | {M}pb(I) |
| Symmetric session key shared by *I* and *R* | SessionKey |
| Symmetric key encryption | {M}k(I,R) |
| Symmetric key shared by *I* and *R* | k(I,R) |
| 3_leak. C -> : T1 | C leaks T1 to the public |

**Table 5.1:** Syntax when modelling cryptographic primitives in SPDL [Cre08]. Since scyther-proof is built on scyther, they share syntax for modelling protocols.

In the modelling, focus is kept on the cryptographic core of the protocols, and other elements are abstracted away. The excerpts of the protocol models listed in this chapter are also slightly simplified compared to those run in scyther-proof. The complete protocol models are listed in Appendix C. Compared to the original models, T1start and T1expire are here abbreviated T1sta and T1exp. Furthermore, Author*i* elements are abbreviated Auth*i*. Similarly to chapter 4 a CC is represented as $C$, KDCs as $A$, and E-nodes as $E$. The CCs home KDC, HKDC, is left out of the model as it is only statically involved by issuing a service statement prior to the protocol.

### 5.1.1 Protocol model A

Following is an explanation of the modelling of Protocol $A$ from Figure 4.6 in SPDL. The Kerberos part of protocol modelled here is slightly changed from the Kerberos model that is available on the github page of scyther-proof[1]. It is changed to a four-pass model, and additionally embeds protocol $a$ from Figure 4.4 within its messages. The timestamp $L1$ stating the current time of the client is modelled as a nonce leaked to the adversary before the protocol is run. Furthermore, $N1$ is a nonce, $T1sta$, and $T1exp$ are timestamps stating the start and expiration time of the ticket. These are similarly to $L1$ modelled as nonces leaked to the adversary. The session key for $C$ and $E$, called $K_{CE}$, is modelled as $SerKey$, short for server key.

In Listing 5.1, $C$ includes in the first request a service statement representing its request for service within the first Kerberos message. Contained within are the elements $Auth2$ and $Auth1$ which represent service statement from the $CC$, and $HKDC$ respectively. In the return message, a third element $Auth3$ is included in the service statement issued by $A$.

**Listing 5.1:** The first two messages of protocol $A$ are displayed below. Note that the signature over the service attributes issued by HKDC, $Auth1$, is removed.

```
1_leak. C-> : L1
1. C -> A : C,S,N1,L1,{Auth2,Auth1,C,L1}sk(C)

2_leak. A -> : T1sta,T1exp
2. C  <- A: C,{C,E,SerKey,T1sta,T1exp}k(A,E),
{S,SerKey,T1sta,T1exp,N1,{Auth3,A,C,T1sta,T1exp}sk(A)}k(C,A)
```

Listing 5.2 shows that in the message from $C$ to $E$, the service statement from $A$ is included by $C$ under a new signature. Entity $C$ has made a choice based on $A$'s decision, called $Auth4$, which should be a subset of $Auth3$. Finally, in the last message $E$ has made a decision $Auth5$, signed it and sent it to $C$ in the final message.

[1]https://github.com/meiersi/scyther-proof

**Listing 5.2:** The last two messages of protocol *A* are displayed below. *C* includes its service statement in the Kerberos authenticator when approaching *E*

```
3_leak. C -> : T1
3. C  -> E: A,{C,E,SerKey,T1sta,T1exp}k(A,E),
{C,T1,{Auth4,{Auth3,A,C,T1sta,T1exp}sk(A),C,T1}sk(C)}SerKey

4_leak. E -> : T2
4. E -> C : {T2,{Auth5,E,C,T2}sk(E)}SerKey
```

### 5.1.2 Protocol model B

Following is an explanation of the modelling of protocol *B* from chapter 4. In Listing 5.3 the AS-exchange of Kerberos is modelled with the PKINIT specific extension. The service statements are included directly within the Kerberos messages.

**Listing 5.3:** The first message of protocol *B* is displayed below. A hash function is applied to the entire original request, and signed by *C* to ensure its authenticity.

```
1_leak. C-> : L1
1. C -> A: C,E,N1,L1,{C,A,NC,Auth1,Auth2,L1}sk(C),
{L1,N1,h(C,E,N1,L1,{C,A,NC,Auth1,Auth2,L1}sk(C))}sk(C)
```

In Listing 5.4, the authenticator is encrypted with a session key, *SesKey*, rather than with a long term key shared between *C* and *A*. Entity *C* obtains *SesKey* by decrypting the PKINIT response with its private key. The entire request initially sent by *C* is returned authenticated with a MAC keyed with *SesKey*. Observe that the nonce sent by *C* in the first request is returned by *A* in the response.

**Listing 5.4:** The second message of protocol *B* is displayed in the listing below. The first two lines are Kerberos AS-REP elements, the last line a PKINIT extension.

```
2_leak. A -> : T1sta,T1exp
2.C <- A: C,{C,E,SerKey,T1sta,T1exp}k(A,E),

{E,SerKey,T1sta,T1exp,N1,{A,C,NC,Auth3,T1sta,T1exp}sk(A)}SesKey,

{{SesKey,h(SesKey,(C,E,N1,L1,{C,A,NC,Auth1,Auth2,L1}sk(C),{L1,N1,
h(C,S,N1,L1,{C,A,NC,Auth1,Auth2,L1}sk(C))}sk(C)))}sk(A)}pk(C)
```

The last two messages of protocol *B* are included in Appendix C. They are similar to those displayed in Listing 5.2, with the exception of how the service statement received from *A* is treated, and the inclusion of a nonce $NC2$ instead of a timestamp. There, entity *C* has received service attributes $Auth3$ from *A*, but instead of sending this element directly to *E*, its own decision $AuthX$ is included under the signature with $Auth4$. Finally, the server *E* returns the nonce $NC2$, and its final decision on service $Auth5$ under a signature to *C*.

## 5.2 Security Goals

From a practical point of view, one of the most important goals of the new protocols is that $C$ receives assurance from $E$ and $A$ on agreed policy, i.e. that it achieves non-injective agreement on exchanged data. For $E$, agreement with $A$ and $C$ on negotiated service attributes is the most important goal, as the final decision on service for $C$ is based on these data. However, it might not need to agree with $C$ and $A$ on all data elements in the protocol, as long as identities, keys, and service policy is correctly interpreted. Hence weak agreement might suffice. For $E$, it might also be relevant to determine if $C$ actually did communicate with $A$ before approaching $E$. For that reason, ordering of events will be studied. PKINIT Kerberos and standard six-pass Kerberos were formally verified using scyther at Cryptographic Protocol Verification Portal (CPVP)[2]. For Kerberos they verify weak agreement for $A$, $C$, $G$, and $S$. $G$ being a ticket granting server, and $S$ an application server. PKINIT is modelled with the same security goals, except that secrecy of session key $K_{CA}$ is included. In PKINIT, non-injective agreement holds for entity $C$. A generalized version of public key NSL was theoretically proven in [Cre06] to possess injective synchronization. Following the disjoint encryption theorem [GT00], the same security goals should hold for the composed protocols.

## 5.3 Setup of experiment

To make the experiment reproducible, the setup of scyther-proof as used in the thesis is explained here. The experiment was performed on a computer with Intel(R) Core(TM) i7-4710HQ CPU at 2.50 GHz, and available RAM was 7,89 GB RAM. Operating system was Windows 10 version 1511, OS build 10586.104, 64 bit operating system with x64-based processor. On it, VirtualBox version 5.0.16 r 105871 was downloaded from https://www.virtualbox.org and installed according to the guidelines on the website. A new virtual machine was set up and assigned the following resources from the host machine: 5028MB RAM, and one core from the CPU. It was also assigned 40 GB of disk storage. An Ubuntu 14.04 64-bit image was downloaded from http://www.ubuntu.com/desktop, and the virtual machine was started from that image. Next, the source code to scyther-proof was found at the GitHub account of its main developer Simon Meier, and download by running (1). The second step was to install Haskell using (2). Finally, step into the root directory of downloaded scyther-proof and install the cabal tool by running (3).

```
(1) $ wget https://github.com/meiersi/scyther-proof
(2) $ sudo apt-get install haskell-platform
(3) $ cabal install
```

[2]http://crypto-protocol.nict.go.jp/about_us.html

## 5.4 Results

After installing the tool as described above, scyther-proof was run with the *shortest*-flag, and with the protocol models and goals as input. That is, the proof with the minimum number of applications of the chain rule was found. Details can be found in [Mei13]. Security goals successfully proved with the protocol models of section 5.1 are listed here. The raw proof output by scyther-proof is included in Appendix C. Table 5.2 define the most common syntax for modelling security properties.

| Explanations of security properties | |
|---|---|
| secret(A,2,K,{C,A}) | If role $A$ has executed step 2, then $K$ is secret to anyone but $C$ and $A$. |
| step(2, A_2) | An instances of A running in thread 2 has completed step 2 of the role. |
| uncompromised(C#2) | The value of C variable is uncompromised in thread 2. |
| role(2) = A | An instance of role A is running in the second thread. |
| C#2 = C#1 | Thread 1 and 2 agree on the value of the C variable. |
| St(2,A_2) < St(1,C_2) | An instance of role A running in thread 2 completed step 2 before an instance of role C running in thread 1 completed step 2. |

**Table 5.2:** Syntax when modelling security properties in scyther-proof [Mei13]. Agreement on variables, ordering of events in time, or specific premises assumed.

### 5.4.1 Protocol A

Contrary to Kerberos, role $A$ in thread 2 here receives the values $C$, $Auth1$, $Auth2$, and $L1$ authenticated by $C$ in thread 1. These items are included under the signature, $sign_{CC}(Auth1, Auth2, L1, C)$, received from $C$. Since these values are authenticated under a signature created by $C$, $A$ agrees with $C$ on these values. In standard Kerberos, entity $A$ does not know who sent these data items, and therefore cannot deduce that $C$ sent these items before they were received by $A$. Assume that $A$ is running in thread 2, has completed step 2, and that $C$, $A$, and $E$ are uncompromised. The properties in Listing 5.5 are then proved for an entity $A$.

**Listing 5.5:** Authentication for role $A$ in protocol A is illustrated below. The reader is referred to Table 5.2 for details on the syntax.

```
C#2 = C#1    &    Auth1#2 = Auth1#1 &
L1#2 = L1#1 &    Auth2#2 = Auth2#1 &    St(2, A_1) < St(2, A_2)
```

Assume that $E$ is in thread 3, has completed step 4, and that $C$,$A$, and $E$ are all uncompromised. It implies for $E$ that there are two other threads running, 1 and 2.

Role $A$ is running in thread 2, and role $C$ is running in thread 1. Furthermore, $E$ agrees with $A$ on the following statements: that $A$ is running in thread 2, and on the values $C$, $A$, $E$, $T1sta$, $T1exp$, $SerKey$, and $Auth3$. $E$ agrees with $C$ that $C$ is running in thread 1, and on values $C$, $A$, $E$, $T1$, $T1sta$, $T1exp$, $SerKey$, $Auth3$, and $Auth4$. Concerning causality, $E$ determines that $A$'s send happened before $C$'s receive, which in turn happened before $C$'s second send, and finally $E$'s own receive.

```
St(2,A_2) < St(1,C_2) < St(1,C_3) < St(3,S_3)
```

Assume role $C$ is running in thread 1, that $C$, $A$, and $E$ are uncompromised, and that $C$ has completed step 4. It implies for role $C$ that there exist two other threads, 2 and 3, with role $A$ running in thread 2 and role $E$ running in thread 3. Furthermore, $C$ agrees with $A$ and $E$ respectively on the following items:

```
role(2) = A            &        role(3) = E           &
A#1 = A#2              &        C#1  = C#3            &
C#1 = C#2              &        E#1  = E#3            &
T1sta#1 = T1sta#2      &        T2#1 = T2#3           &
T1exp#1 = T1exp#2      &        Auth5#1 = Auth5#3     &
Auth3#1 = Auth3#2      &        T1sta#1 = T1sta#3     &
                       &        T1exp#1 = T1exp#3     &
                                SerKey#1 = SerKey#3 &
```

Concerning causality, as $C$ is involved in all send events and receives corresponding answers to these, it is able to determine a strict ordering of all send and receive events.

```
St(1, C_1) < St(2, A_1) < St(2, A_2) < St(1, C_2) <
St(1, C_3) < St(3, S_3) < St(3, S_4) < St(1, C_4)
```

### 5.4.2 Protocol B

The same secrecy goals proven in protocol $A$ also hold for protocol $B$. The only difference being that secrecy of $SesKey$ was proven for $A$ and $C$. Regarding authentication, assume role a $C$ running in thread 1, which has completed step 2 of the protocol, and that $C$, $A$, and $E$ are all uncompromised. It implies for $C$ that there exists a role $A$ running in thread 2, and that $C$ agrees with $A$ on the following data items: $A$, $C$, $NC$, $Auth1$, $Auth2$, $Auth3$, $L1$, $T1sta$, $T1exp$. Since $A$ included in the response message the nonce sent by $C$, $C$ can conclude that $A$ did not send its response before it received the nonce sent by $C$ in the first message.

```
St(1, C_1) < St(2, A_1) < St(2, A_2) < St(1, C_2) <
St(1, C_1) < St(1, C_2) < St(2, A_1) < St(2, A_2)
```

In the challenge-response between $C$ and $E$, much of the same properties hold as in the exchange between $C$ and $A$ above. Assume $C$ is running in tread 1, has completed step 4 of the protocol, and that $C$, $A$, $E$ are uncompromised. Then, $C$ is able to conclude that there is a role $E$ executing thread 3, and they further agree on the following data items: $E$, $C$, $NC2$, $T2$, $AuthX$, $Auth4$, $Auth5$. Full ordering of events for $C$ is also deduced.

For a role $A$ running in thread 2 which has completed step 2 with $C$, $A$, and $E$ uncompromised, it is able to determine that there exists a role $C$ running in thread 1. Furthermore, role $A$ agrees with role $C$ on the following data items: $C$, $NC$, $Auth1$, $Auth2$, $N1$, $L1$. Here, role $A$ is able to conclude that the first message was in fact sent by role $C$, and hence can conclude that it responds to $C$'s request in the second message.

```
St(1, C_1) < St(2, A_1) < St(2, A_1) < St(2, A_2)
```

Finally, for role $E$ of protocol $B$, in addition to agreement on the items mentioned in protocol $A$, the nonces used within protocol $b$ are agreed upon.

```
role(2) = A                      &    role(1) = C                          &
NC#3 = NC#2                      &    T1exp#3 = T1exp#1                    &
T1exp#3 = T1exp#2                &    AuthX#3 = Auth3#1                    &
T1sta#3 = T1sta#2                &    Auth4#3 = Auth4#1                    &
SerKey#3 = SerKey#2              &    T1sta#3 = T1sta#1                    &
C#3 = C#2 & A#3 = A#2            &    SerKey#3 = SerKey#1                  &
E#3 = E#2 & NC#3 = NC#2          &    C#3 = C#1 & A#3 = A#1                &
                                 &    E#3  = E#1 & NC#3 = NC#1             &
                                 &    NC2#3 = NC2#1 & T1#3 = T1#1
```

Role $E$ is also able to conclude that $A$ received its first message from $C$. Role $C$ is therefore almost able to determine a full causality chain.

```
St(1,C_1) < St(2,A_1) < St(2,A_2) < St(1,C_2) <
St(1,C_3) < St(3,S_3) < St(3,S_4)
```

The protocols defined in chapter 4 were formally analyzed in this chapter. Modelling of protocols and security goals were done in SPDL, and proven correct using scyther-proof. In the next chapter, the results obtained will be discussed in addition to a discussion of the result in the context of PCN.

# Chapter 6 Discussion

There are two main parts to this discussion. First, the proposed protocols and results from their analysis are discussed with emphasis on limitations and trustworthiness. Second, the proposed solution is discussed within context of practical deployment with a special focus on IDPKC. The chapter is finalized with the authors opinion of the proposed solutions for entity authentication at the PCN-2 interface.

## 6.1 Discussion of results

The following section will discuss architectural choices made in chapter 2 and chapter 4, before variations of the Kerberos protocol. Next, the extent and limitations of the results obtained in chapter 5 are discussed with emphasis on security goals, adversary model, protocol models, and scyther-proof.

### 6.1.1 Architectural decisions

Three architectures of authentication were considered for PCN. A strict peer-to-peer model ($X$), a central AAA-server aided model ($Y$), and a token-based model ($Z$).

IDPKC IKE was identified as one alternative of $X$. In IKE, the identities of authorized entities are often pre-stored in router policy. For example, a range of IP-addresses, or domain names found through reverse Domain Name System (DNS) could be assigned a specific policy. It was found that in IKE based on IDPKC, the ID-field of IKE could in principle be removed if global network addresses are used for identification. However, [PCN10] states that IP-addresses shall be dynamically assigned connecting CCs in PCN. As transaction authentication on network layer was assumed, media access control addresses are not present for identification. Media access control addresses are unique identifications assigned to network equipment by manufactures. These are often used for identification at link-layer of the OSI stack. One exception being IPv6 link local addresses, which are constructed from an entity's mac-address. These are however only globally unique within a given link domain [Dee98]. IDPKC IKE does not offer a simple way of authenticating CCs from several

PCSs. Storing service policy in E-nodes corresponding to CCs from several PCSs, clearly does not scale well.

To simplify the authentication of CCs coming from other PCSs, model $Y$ could be considered. A common way of separating entity authentication from transaction authentication is through the use of Extensible Authentication Protocol (EAP) [AE08]. EAP is an application layer protocol enabling mutual entity authentication between a client and a central AAA-server. Not being able to process entity authentication, a router forwards EAP authentication requests through a RADIUS [WRRS00] or diameter [FALZ12] protocol to an AAA-server. The setup is illustrated in Figure 6.1. Using a central server for authentication and authorization fulfills the *scalability* requirement from chapter 2. One limitation is that the router did not participate in entity authentication. Instead, a client authenticates an AAA-server which tells the client to trust the router. In principle, one legitimate router could masquerade as another legitimate one, without the client being able to tell the difference [EA04].

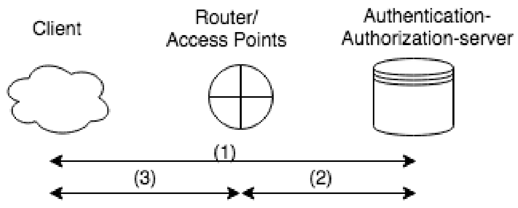

**Figure 6.1:** Figure showing a client participating in entity authentication with an AAA-server (1). Keying material is then transported to router (2), and client (3).

Commonly known standards that use centralized authentication are IEEE 802.11 Wireless LANs and cellular networks (GSM/UMTS/LTE) [AE08]. In IEEE 802.11 [A+03], the authentication and key establishment through EAP is followed by transportation of the shared key from an AAA-server to an access point. Due to how cellular network towers and wireless LAN access points are deployed, the risk of compromise has been justified by external considerations [EA04]. In PCN terminology, the client corresponds to a CC, and the router an E-node. To transport service policy, one option is to pull policy corresponding to a given identity from a CCs home PCS. However, if continued connectivity from visitor PCS to home PCS is difficult to assure, a CC might not receive the necessary services in time.

To solve the problem of varied connectivity, model $Z$ was suggested. That is, a push-based model for authentication based on tokens. By using tokens, a TTP-approach is assumed. In a TTP-approach, one entity functions as a bridge between two other entities where each of them trusts the TTP. One argument for the use of a $Z$ architecture contrary to $Y$, is that an E-node should be able to temper intended service delivery based on local considerations. Instead of simply receiving at list of services the CCs are authorized to use, the E-node can modify this decision. Either based on the strength of authentication of a CC, or on network properties of the PCS, similarly to SESAME [KPP94]. In [Ash97], Ashley argues that a common disadvantage of push- contrary to pull-models is that the access rights stated in the ticket might not be fresh. To remedy the situation, authentication between CC and E-node could be assumed to depend on a TTP regularly issuing fresh service rights.

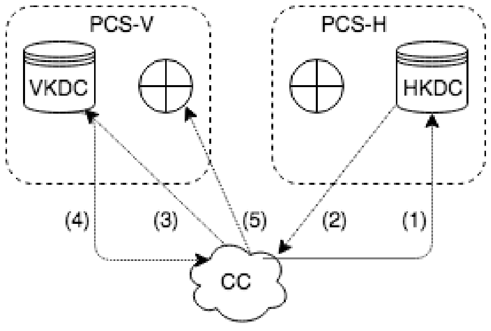

**Figure 6.2:** A token-based architecture for authentication. Continued connectivity between PCSs is not necessary for connection setup between a CC and an E-node.

In the protocols in chapter 4, VKDC functions as a TTP between E-nodes and CCs. Similarly, HKDC functions as a TTP between VKDC and CC. One could assume that VKDC pulls service rights from HKDC when connectivity is established. In between pull requests, a push-based model could be used based on the issued service statement. To limit the validity of service statements, timestamps were included. A CC would then have to approach the VKDC for a new service statement after a given time, and obtain its updated service rights retrieved by VKDC from HKDC. The setup is illustrated in Figure 6.2. A CC first requests its HKDC for a statement on which services it is authorized to use (1)(2). When connecting to a foreign PCS, the CC push this statement to VKDC (3). After authenticating the issuing authority (HKDC), and making a decision on authorized service in PCS-V, VKDC responds with a new service statement (4). This document states a CCs authorized services within PCS-V. Finally, the PCS-V service statement is pushed to E-nodes the CC connects to. These ideas aid in fulfillment of the requirements *scalability* and *efficiency*. However, the requirements of interoperability and security were not sufficiently covered. Focus was then shifted towards token-based solutions specified as Internet standards. Specifically, the Kerberos protocol with its KINK extension to set up SAs for IPSec were a natural choice for further study.

### 6.1.2 Kerberos in MANET-like networks

In [PM04] changes where suggested to the traditional Kerberos v5 [NHYR05], in order to make it more suitable for MANETs. In addition to replication of KDCs to distribute authentication, Kerberos Assisted Authentication in Mobile Ad-Hoc Networks (KAMAN) build on four-pass Kerberos [CKM00]. Later, a modified version of KAMAN intended to prevent against replay attacks was proposed in [BK12]. The Modified Kerberos Assisted Authentication in Mobile Ad-Hoc Networks (mKAMAN), used the fact that user passwords were hashed in order to encrypt the entire packet to protect against replay. In mKAMAN, two clients want to communicate with the help of a KDC. In the equivalent of the AS-exchange in Kerberos V5, the KDC encrypts the packet with the first client's hashed password. Inside the packet, and together with the ticket for the second client, is the hashed password for the second client. The first client is able to encrypt the entire packet going to the second client with its the hashed password received from KDC.

Even though mKAMAN presents a way of using Kerberos in MANET-like environments, it is not straight forward to extend its use across security domains. To use a variation of Kerberos in coalition military operations, two choices were identified; cross domain tickets and public key cryptography. A choice was made to follow the latter approach, as it might be difficult for a CC to know which PCSs it will visit in advance, and hence needs to request cross domain ticket for. Nevertheless, the idea of KAMAN to use four-pass Kerberos instead of standard six-pass was pursued. The reasoning behind this choice was that TGSs were specified in Kerberos V5 to not expose the user's password more than once [NHYR05]. This is however no concern when public key cryptography is used for initial authentication.

PKINIT was chosen for initial authentication in Kerberos. Alternatives to PKINIT as public key cryptography in Kerberos have been created. Examples are Public Key Kerberos for Distributed Authentication (PKDA) [SC97], Public Key Utilizing Tickets for Application Servers (PKTAPP) [MHN97], and Public Key Cryptography for Cross-Realm Authentication in Kerberos (PKCROSS)[HTTm01]. In PKDA, a client authenticates directly to an application server using public key cryptography, and the server issues a ticket for itself. Workload is then distributed away from the KDC. The need for an online key distribution center is thus eliminated. PKTAPP tries to achieve the goals of PKDA using PKINIT. With PKTAPP, a client performs the AS-exchange using PKINIT with a Local Ticket Granting Server (LTGS) at the application server. When a client sends an AS-REQ to a LTGS, it also specifies an application server in order to get a service ticket. Instead of receiving a TGT as in traditional Kerberos, the client receives a service ticket. Another difference between PKDA and PKTAPP is that PKDA requires the client to compute the symmetric key, while with PKTAPP it is the responsibility of the server. A comparative study of the three protocols was performed in [AJR11], and the performance of public key versions of Kerberos in large networks was analyzed in [HM01]. When cross domain authentication is necessary, [HM01] concluded that PKCROSS is more efficient than PKTAPP if a client authenticates to several entities in the visiting domain. However, PKCROSS requires continued connectivity between security domains, and is therefore not considered further. Neither is using a variation of PKDA such a PKTAPP, as authenticating directly an E-node would scale similarly to architecture $X$ discussed previously.

Authentication in Kerberos between $A$ and $B$ depends on a $TTP$. A consequence of the TTP-model is that $A$ does not authenticate $B$ independently of TTP. Since the thesis assumes a CC authenticating with attributes linked to service classes, not authenticating the E-node separately is equivalent of not receiving assurance from E-node on agreed service level. To fulfill the *security* requirement, a separate authentication of E-nodes to CCs was believed necessary. The experiment proving security properties of the composed protocols are discussed in following sections.

### 6.1.3 Security goals

In protocol $A$, entity $A$ was not able to conclude anything about $C$ from data received in the clear. Since an active intruder was assumed in the DY-model, $A$ may not even known if $C$ is present on the network, as an active intruder could impersonate $C$. Hence it does not even achieve recent aliveness from the unprotected data. The service statement was signed by $C$ over freshly generated data item, and hence $A$ received assurance that these data items was recently sent by $C$. Assuming synchronized clocks, and that $VKDC$ is familiar with $C$ in advance, $VKDC$ was from these items able to verify the identity of $C$, and verify the freshness of this message. Another important point is that $A$, $C$, and $E$ all agree on the $Auth3$-element. The consequence of which is that all three entities have a common understanding of the PCS's offered services. The other security goals of protocol $A$ related to agreement are trivial in the sense that only an entity with knowledge of a specific key is able to construct a signature or encrypt data. As a non-compromising adversary is assumed, only legitimate entities are in possession of these keys. When an entity receives data protected with such a key, it will be able to conclude who constructed the message and hence achieve agreement over the protected data items. However, as neither $A$, nor $E$ are involved in all messages, they only achieve agreement on some of the values, and hence weak agreement is concluded. Related to ordering of events, it is no surprise that $C$ is able to order all events, and achieve non-injective synchronization, as it was involved in all send and receive events. When receiving message 2 from $VKDC$, $C$ was able to tell if an adversary had been active and changed values even though not all were cryptographically protected. The reason is that $C$ was the one sending these unprotected data items, and can therefore compare against the response received from $A$ to see if an adversary changed the values.

For protocol $B$ the introduction of public key cryptography in the first message gives $A$ assurance on all data items sent from $C$. Related to ordering of events, an assumption is made on synchronized clocks, and $A$ was then able to conclude freshness from $C$'s current time $L1$. For $C$, the first received message contains a nonce sent by itself in the first sent message, which enables $C$ to deduce that this message is not a replay. Interestingly, $C$'s independent signing of the service statement may be unnecessary in the first exchange as the data is already signed as part of PKINIT, and nonces are used for replay protection. One exception may be if the service statement is forwarded from the E-node to another entity, while the PKINIT exchange stops at the E-node. The CC could then benefit from assurance on whom conducted the authentication, and hence created the signature over the returned nonce. In terms of replay protection, the effect of protocol $b$ is quite different in the second exchange. In an attempt to model the effect of unsynchronized clocks, the timestamps were removed and only the service statement was replay protected. Then, the property of non-injective synchronization for $C$ did still hold. That is, replay protection using nonces in the inner authentication protocol also provided the composed protocol $B$ with replay protection. In addition to inclusion of nonces, protocol $b$ inside $B$ was modelled slight different than protocol $a$ inside $A$. Contrary to protocol $a$ where $C$ included the entire authenticated service statement from $A$ in the request to $E$, in protocol $b$ entity $C$ just copied some of the information. It is therefore modelled as $C$ sending $AuthX$ to $E$, instead of forwarding the entire $Auth3$. The effect is that $E$ does not achieve agreement with $A$ on $Auth3$. The solution used in protocol $b$ may be more applicable when $A$ is only giving advice to $C$ on service policy, rather than sending commands on policy for the PCS.

### 6.1.4 Adversary model

Assuming perfect cryptography may be considered controversial when analyzing protocols, as the threat of cryptanalysis is assumed negligible. Even though a given cryptographic scheme might not break completely, small pieces of information could be leaked by cryptanalysis and utilized in an attack on other properties of the system. Such a threat may be even more realistic if the adversary is able to predict the messages being signed, such as when timestamps are used, and when the public key is known to the adversary in advance. A similar limitation can be made on the assumption of non-compromising adversaries. Even though an adversary may not compromise a CC or E-node entirely in the physical sense, sensitive information could be leaked inadvertently. Specially crafted data packets sent from the adversary to an PCN-entity could trigger an uncontrolled response, possibly leaking useful control data. In that sense, the DY-model can be considered quite static, and modelling the protocols with dynamically compromising adversaries could be necessary for a more realistic analysis. This is identified as a topic for further work in chapter 7.

### 6.1.5 Symbolic model of cryptography

In the protocol models of chapter 5, cryptography was treated as black boxes. However, not all black-boxes in cryptography are alike. The type of black-box used, and its input-output values will depend on the cryptography inside.

In the suggested protocols, only the separate authentication protocol inside Kerberos was assumed to be based on IDPKC. A signature scheme only requiring a recipient to know the identity of the signer in advance can be assumed. With IDPKC based on pairings, the situation is different. Auxiliary parameters must then be extracted by the sender from the recipient's identity in order to encrypt properly, and hence requires a different black-box to model properly. The question if such constructions could be used effectively in PCN is briefly discussed in section 6.2. Another consideration when modelling the protocols in chapter 5 was the public key in the IDPKC setting. As the public key and the attributes essentially are the same, a choice was made to model the public key as attributes Auth$i$, and not as $pub_i$. In some situations, privacy with regard to what a specific entity is signing might be a requirement. To avoid eavesdroppers trying specific public keys to verify some signature, a user may want the public key to be indistinguishable from random. By modelling a public key as an attribute the opposite of privacy, namely assurance that the other entity knows the data items can be modelled. The extraction of the public key from attributes is not explicitly modelled in this thesis. The public key corresponding to the private used with the signatures is therefore assumed known to the recipient in advance. Theoretically, one possible reason for including the *parameter-extraction* algorithm would have been to model if an adversary could perform a man-in-the-middle attack by registering the public key as its own at the KGC. Since private keys are assumed stored locally at PCN-entities in advance, no online KGC is necessary. Additionally, in a military setting it would be reasonable to assume strict control on the distribution of private keys.

### 6.1.6 Scyther-proof theorem prover

One clear disadvantage of scyther-proof is the modelling of time. In the experiment here, time is modelled as fresh numbers leaked to the adversary at the beginning of a session. Therefore, it might be difficult to make brunt statements about replay protection, and the difference between fresh nonces, timestamps and sequence numbers. Especially one observation during the experiment was noteworthy. In an attempt to model the behavior when timestamps were not present, a nonce under a signature in protocol $B$ seemed to function as a sequence number. That is, the ordering of events for $E$ was still proven correct when the nonce within the signature was present, and the timestamps were not. As nonces are random numbers not previously known to $E$, it should not be able to make a deduction about timing when it did not have any local state to compare the nonce with. To assume shared state between $C$ and $E$ may be an unreasonable assumption. The limited modelling of time also constitutes a challenge when assumptions of clock skew and their impact on $T1sta$ and $T1exp$ need to be made. On the other hand, versions of Kerberos have been formally analyzed extensively in the past, and focus in the experiment here is not on the Kerberos protocol itself. Rather, focus has been on the agreement of service policy between peers, and the interaction between Kerberos and the inner authentication protocol. Related to the challenge-response mechanisms, $C$ should theoretically be able to conclude injectivity of the response as it can check that it has not been received previously. However, the notion of injectivity is closely related to modelling of time, and is not supported by scyther-proof. Modelling the interaction between timestamps, sequence number and fresh nonces is identified as topic for further work in chapter 7.

The use of formal methods is not a bulletproof methodology. In [BCM13] Basin et.al. formally analyzed the security of the ISO/IEC 9798 protocols for entity authentication. Using the tool scyther, they identified a number of vulnerabilities in the standard related to role-mix up. That was however not the first time the standard has been changed due to new attacks being found [CM10]. As abstractions are used when modelling protocols, weaknesses are sometimes found in later models of a given protocol. Another example being Kerberos with its public key version PKINIT which was formally analyzed in [Tsa08] using Backes-Pfitzmann-Waidner (BPW) formalism. New vulnerabilities were found even though PKINIT had been extensively analyzed prior to that experiment. It is therefore difficult to conclude much about the verified protocols that are not included within the analysis. It also seems difficult to predict which parts of a protocol, or the environment, can be abstracted away without affecting the security analysis.

The next section will contain considerations of practical deployment. Specifically, some elements that are abstracted away from the formal modelling, such as cryptography, will be devoted some time.

## 6.2 Discussion of practical deployment

Previously in the thesis, cryptographic primitives have been treated as black-boxes. The symbolic model of protocol analysis treats cryptographic primitives as having perfect security. Now the black-boxes are opened, and both security related and non-security related issues are discussed.

### 6.2.1 IDPKC in PCN

The study of IDPKC in PCN was interesting for mainly for three reasons. Commonly acknowledges drawback are less prominent, the lack of certificates, and that key distribution is simplified.

First, a commonly argued drawback of IDPKC is that a KGC theoretically is able to store copies of USKs derived from a MSK. In some application, a compromise of MSK can result in an attacker being able to decrypt all information transferred protected with keys derived from this key. However, PCore does not offer confidentiality services, and hence a potential loss of forward secrecy is minimized. In the case of identification schemes or digital signature schemes, a KGC will theoretically be able to impersonate PCN entities. As CCs and E-nodes in PCN are assumed part of a military hierarchy, their privacy with respect to a centralized authority in their home PCS can be questioned. Although impersonation of PCN entities can cause issues for accounting services, it is more of a policy problem rather than security. If a KGC is physically secured, key escrow might be a tolerable risk. As mentioned in [TD07], it is theoretically possible to destroy the KGC after deployment if all nodes are configured prior to use. Note that the KGC would be needed in the case of renewing or regenerating private keys [TD07]. If rekeying is not possible, long lifetimes would be necessary for all private keys. Regarding arguments of single point-of-failure, several constructions to distribute the KGC based on threshold cryptography has been suggested [BF01].

Second, while certificate-based PKIs are widely deployed and used on the Internet, such as in Transport Layer Security (TLS) [Die08], the complex certificate handling has been subject to much discussion [ES00]. One of the most significant disadvantages of using digital signatures instead of MACs within tickets and authenticators, is the increased load from certificates. When each signature requires an additional signature in the form of a certificate, the load on bandwidth resources quickly increases. However, the advantage of digital signatures over MACs is relevant when forwarding tickets. When an E-node receives an element from VKDC through the CC, it should either be included in a ticket issued under a key the CC does not know, or be authenticated using a digital signature. The approach based on signatures was chosen because one of the goals of the thesis was to give CCs increased control over expected service delivery. For that reason, it was necessary for a CC to be able to verify the authenticity of a VKDC's proposal. The use of signatures, as opposed to MACs, enables a separation of protocol $a$ and $b$ from the Kerberos variants since symmetric keys are not needed for the authenticity of service attributes. Such a separation is not fully taken advantage of in this thesis, but could prove valuable in other scenarios.

The third reason is related to key distribution, for both private and public keys. In certificate-based approaches the private key never leaves the user, and hence is less exposed. In IDPKC however, private keys must be securely transferred to entities from KGC, which is considered a possible weak link in terms of security. In the context of PCN, it is assumed that each PCS would have a KGC which performs *parameter-generation algorithm* and *key-extraction algorithm* before deployment. When setting up the system, the KGC would verify the authenticity of an entity before securely storing the private key in a tamper resistant card. MPKs are also assumed stored in tamper resistant cards to protect their authenticity. Nevertheless, if a private key is exposed during use, it would have to be invalidated. As a protocol for spontaneously updating private keys increases complexity, one could assume a policy where private keys are changed upon maintenance of PCN-entities at regular time intervals. For this reason, it is not a good idea to use the attributes directly as public keys. An extra string should be added to the chosen attributes so that they can be updated without changing the format of public keys. Additionally, it is often recommended to include a timestamp as part of the public key so it is invalidated after a given period.

Regarding distribution of public keys, these are in IDPKC often assumed known to the verifier in advance. In the protocols from chapter 4, we get one identity for free, that is the Kerberos principal name. Additionally, a predefined set of service attributes can be assumed stored in each entity. If one assumes public/private key pairs shared within a group, authentication may be performed just based on a set of attributes shared within a group. Alternatively, the Kerberos-principal identity may be included as part of a global identity for an entity together with attributes and other auxiliary data in order to create a unique identity. An E-node could then receive the Kerberos-principal identity for a CC in the standard way through the service ticket from the local KDC. A CC's public key would be reconstructed using the Kerberos-principal identity and a set of attributes provided by the CC. The advantage being that an E-node could link a CC's service usage to a specific identity, and not just to a group of entities sharing attribute keys.

### 6.2.2 IDPKC Signature schemes in PCN

Several IBS-schemes can be realized using number theoretic primitives. Additionally, the use of *pairings* on elliptic curves has led to signature schemes with new functionality. However, these often come at the cost of much higher computational load. Focus of this section is on efficiency and security considerations of IDPKC. Two schemes based on number theory are discussed first, before a brief discussion on pairing-based signatures in general.

#### Service class through a single identification scheme

In chapter 3 it was identified that one of the most common ways of constructing identity-based signatures was from identification schemes. Schnorr signature scheme was patented until 2008, but its attractive properties such as efficiency and flexibility has quickly made it attractive for resource constrained environments.

One example being [GG09], in which Galindo and Garcia propose a lightweight IBS schnorr signature scheme. Their construction is based on using two regular schnorr signatures to achieve an id-based equivalent. One signature is computed by

KGC using MSK over the user's identity. This signature is then used as the public key for the second signature. They achieve a scheme with comparable efficiency to the best id-based schemes based on integer factoring, discrete logarithm on elliptic curves, and pairing-based alternatives [GG09]. For use in the protocols proposed in chapter 4, one could associate a set of attributes $A, B, ..., D$ with a service class before deployment. The service class would then constitute the CC's identity, and a corresponding private key would be securely stored in the CC. Depending on the situation, a CC could receive one or more private keys associated with several service classes in advance. The CC's HKDC could then sign the name of the service class with $MSK$ before deployment and use this as a public key for that particular service class. For example, a service class $\phi =$ A $\|B\|...\|D$, for attributes $A, B, ..., D$. The concept of building identities composed of attributes is illustrated in Figure 6.3.

One clear disadvantage of pre-established service classes linked with key pairs is its limited flexibility. A CC would not be able to negotiate on attributes at the VKDC. The VKDC could instead function as a certification authority for a CC connecting to the PCS. After checking the validity of a service statement issued by HKDC, VKDC could copy these attributes to a new service statement for that PCS. Approaching an E-node, the CC could prove possession of its identity, i.e. concatenation of all its attributes. The E-node would then itself determine, based on local policy, if the proved attributes are a super-set of required attributes for a given service. In such a scenario, all the CC's attributes are presented to an E-node, regardless of policy, which could result in a privacy issue. A CC would also need one private key stored locally for each of the pre-established service classes it is authorized to use. An E-node's ability to dynamically change policy based on local considerations would be limited as it cannot ask the CC to authenticate with a subset of attributes.

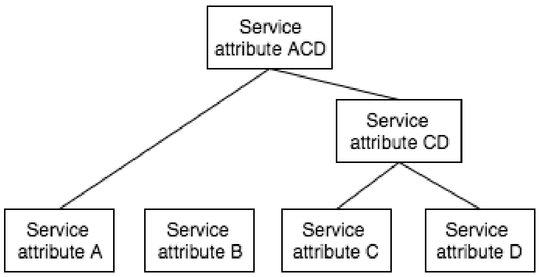

**Figure 6.3:** A hierarchy of service attributes. Base service attributes are named A,B,C, and D. Each attribute corresponds to service policy stored in E-nodes.

### Service classes defined through multiple identification schemes

In a second approach, private keys may be issued for each single attribute. Attributes can then more flexibly be combined to create specific service classes. Different E-nodes could, depending on the environment they are installed in, achieve more specific policies on which attributes are necessary for a given service level. In the simplest case, a CC would then have to prove possession of each private key in parallel identification schemes as illustrated in Figure 6.4. However, in a more advanced case, a CC could create a batch consisting of a set of attribute signatures [GLSY04]. In their generalization of schnorr identification scheme, Gennaro et.al. claim that the bandwidth and computational load of the resulting batch is just above a standard schnorr signature [GLSY04]. They argue that this is an efficient construction when an entity wants to prove possession of several keys at the same time. In [GLSY04] the scheme is proposed for privacy preserving situations of authentication. Theoretically, a CC could have some special attributes which are of sensitive nature, and should only be revealed to certain E-nodes. Contrary to the scheme described above where all attributes are always sent, this scheme allows the CC to adjust which attributes are presented to the network for authentication.

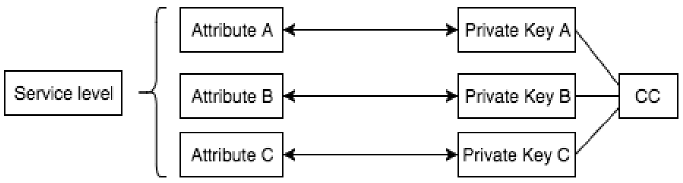

**Figure 6.4:** Identification schemes executed in parallel, one for each attribute corresponding private key. A solution batching these together is presented in [GLSY04]

### Signatures from pairings

In [GPS08] Galbraith et.al. discuss important considerations when using pairings in cryptography. They argue that while purely treating pairings as black boxes is attractive since it abstracts away the complex mathematical background, it can result in schemes having different properties than first believed. With the increased performance of computers, longer keys are needed to provide sufficient security using symmetric key cryptography. As public key cryptography is often used to transport symmetric keys to simplify key management, the strength of the public key primitives must increase accordingly.

In traditional Elliptic Curve Cryptography (ECC), security depends on the difficulty of solving a problem known as Discrete Logarithm Problem (DLP). In pairing-based cryptography, security depends on the hardness of another mathematical problem called Bilinear Diffie-Hellman Problem (BDHP). Although BDHP has been studied extensively in recent years, it is still quite new compared to DLP. Researchers have found that solving DLP leads to solving BDHP, so the security is no stronger than of traditional ECC [MPP$^+$15]. Security of pairing-based cryptography is based on the hardness of finding discrete logarithms in the finite field $\mathbb{F}_{q^k}$. Often $q$ will be a prime or a power of 2 or 3 [KM05]. In [KM05] Koblitz et.al argue that the bit length of $q^k$ must be around the same length as a typical RSA modulus for the same security. This effectively leads to bit lengths of 15360 to match a 256-bit AES key. In traditional ECC, the needed key lengths are decreased compared to RSA when higher security level are required. As this is not the case for pairing-based cryptography, the cost compared to traditional ECC can be significant with high security standards. Details on the required bit-lengths for different security levels are treated in [Len01].

The Boneh-Lynn-Shacham (BLS) signature [BLS04] is based on the Weil pairing, and is considered one of the most efficient uses of pairing-based cryptography due to the short signature lengths. In [KM05] Koblitz et.al. point out that while the signature lengths are shorter in BLS than e.g. Elliptic Curve Digital Signature Algorithm (ECDSA), it needs much longer public keys. Having longer public keys leads to increased cost when computing signatures as the number of exponentiations grows. However, short signatures are not the only application of pairings in IDPKC. Several other constructions such as attribute authentication schemes [YO15], aggregate signatures, and group signature schemes [Pop02] provide interesting functionality. For that reason, a decision to choose pairing-based schemes should be based on the need for a specific functionality not offered elsewhere. A study of these special schemes' potential use in PCN could prove an interesting topic for further work.

The final section of this chapter contains the authors opinion of the proposed solutions.

### 6.2.3 Entity authentication at PCN-2 interface

This section contains the authors opinion. It is outside the scope of the thesis to discuss the applicability of different QoS-schemes in PCN. However, in order to link entity authentication to the fields of QoS and transaction authentication, a short introduction is given. Generally speaking, QoS schemes are divided into schemes that do the signaling in-line as part of the packet header, and schemes that reserve resources in advance. An example of the former is Integrated Services (IntServ) [BCS94] which reserves resources for distinct flows in advance. An example of the latter is Differentiated Services (DiffServ) [BBC+98] which defines per-hop-behavior of aggregate flow. It uses the Differentiated Services Code Point (DSCP) value in the traffic flow field in the IP-header to indicate behavior for a given flow. The interesting idea of authenticating QoS-fields in the IP-header as part of DiffServ was proposed with symmetric keys in [HW08] and asymmetric keys in [CLK05]. These two schemes were briefly introduced in chapter 2. Correctly handling the separation of CCs from PCore through PCN-2 has been the focus of this thesis. Assume a DiffServ approach to QoS enforcement. Assume further that either of the above mentioned solutions to protection of QoS fields in the IP-header is used. Then clearly a huge responsibility rests at the E-node at the edge of a PCS as seen in Figure 6.5. Not being part of the PCore, a CC must trust the E-node to correctly interpret QoS fields in sent packets, re-authenticate, and forward these into the PCore. This was the motivation of increased assurance for a CC on service policy from the E-node.

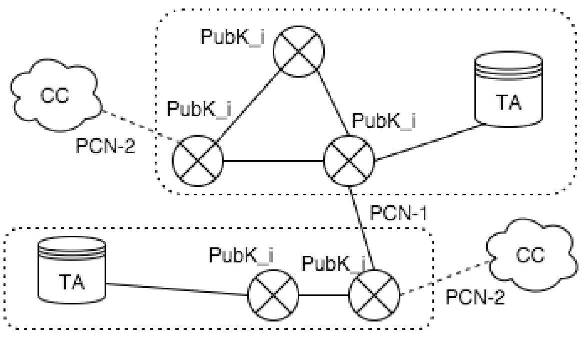

**Figure 6.5:** Packet level authentication using signatures. Distribution of public keys are simplified as they are transferred in line with IP-packets [CLK05].

A TTP token-based approach to authentication was chosen. The drawback is an increased complexity and bandwidth usage at PCN-2 interface. For this reason, the proposed solution does not seem advantageous in tactical environments. One the other hand, the proposed solution does not require the network to pull service policy from the CC's home PCS. In that sense, it might be valuable in more stationary networks, where connectivity can be assured at PCN-2 interface, but may be difficult to guarantee across several PCSs. Intermediate scenarios, such as forward operating bases in more exposed environments may also take advantage of the increased assurance a CC receives from the E-node on service. A relevant question is if the VKDC is necessary at all. Why can not a CC approach an E-node in a visiting PCS with a service token from HKDC? If all E-nodes store public keys of KDCs from multiple PCSs, key distribution is solved. An E-node would authenticate a CC as described in chapter 4. Although, it might then be difficult to rapidly change policy for an entire PCS when situations change. Another argument for a CC approaching the VKDC before E-nodes is to retrieve auxiliary information about the PCS. When a CC connects to a visiting PCS, it may not know much about its offered services. It might therefore be natural to receive such information from a central server before approaching distributed E-nodes.

It has been argued in the thesis that entity authentication based on attributes, rather than on unique identities, is a possible solution to overcome the challenges of authentication in distributed systems. Groups of entities with similar attributes would then be treated as having the same identity. On one side, it provides clear drawbacks with respect to compromised, or ill behaving entities. It may also be difficult to add services for one specific entity after deployment if entities are only identified by attributes shared with others. One the other hand, applying a set of services to an entire group of entities is clearly simplified. There are also advantages of privacy associated with attributes, as it may be more difficult to track specific entities. In the protocols suggested in chapter 4, some form of global identities are assumed. Specifically, the Kerberos principal identities could be used for unique identification of entities when initially authenticating to KDCs. This identity could be used to either block specific entities, or associate some services with a specific entity, but not with other entities in its group. Authentication of a CC by E-nodes could still be performed based on attributes after having obtained a service statement from a KDC. It is the authors opinion that the balanced weight between unique identities and more general attributes, and between distributed and centralized authentication described here, can provide valued flexibility to PCN-2 entity authentication.

Chapter 7

# Conclusions and Further Work

Section 7.1 summarize, and present results obtained, and tries to answer if the objectives of the thesis were fulfilled. Topics for further work are listed in section 7.2.

## 7.1 Conclusions

A pragmatic approach was taken to IDPKC in PCN, by first identifying that a limitation of existing solutions is static service policies. Between different PCSs, and between KDCs, E-nodes, and CCs. An architecture where users push service attributes to the visiting PCS was assumed. Using IDPKC, attributes directly linked to service levels were proposed as a CC's identity. No external mechanism binding an identity and a policy is then needed by the E-node. A CC proves possession of private keys corresponding to one or more service attributes by participating in an identification scheme with the E-node. For interoperability, the attribute-based scheme was proposed carried within four-pass PKINIT and symmetric key Kerberos. The parallel composition of the two protocols was analyzed in the symbolic model using scyther-proof with the DY adversary model. A CC, E-node, and VKDC were able to agree on PCS service policy. Furthermore, a CC received assurance from VKDC and E-node, that its requests for service was interpreted correctly. Due to the fact that military networks often can assume pre-distribution of keying material, and more closed user groups, the commonly known drawbacks of IDPKC are less prominent in PCN than in traditional commercial applications.

PCN was introduced as a way of enabling secure and flexible communications in coalition operations. Specifying interfaces for interconnection of PCSs, flexibility is achieved for CCs to move geographically, and reconnect to PCore elsewhere. The advantage provided by PCN is how PCore can provide high availability service delivery to connected user networks. The thesis has shown that entity authentication based on attributes using identity-based signatures can lead to simplified authentication and increased assurance of service. In effect, easy interconnection of entities with little pre-existing relationship is simplified, enabling the flexibility required by PCN.

## 7.2 Further Work

Two topics for further work are identified. One improving the trustworthiness of the results obtained here, and one extending upon the work done in this thesis.

**Theorem proving:** The Tamarin theorem prover [MSCB13] builds on the scyther-proof tool, and generalizes the underlying theory. It includes support for equational theories and expressive property specification language. Tamarin supports modelling of time, and it could be interesting to study situations where the authentication of service policy and the Kerberos protocol are based on timestamps with different length of validity. Furthermore, its support for dynamically compromising adversaries could enable more realistic modelling of exposed environments. Additionally, Tamarin currently has experimental support for pairing-based primitives and further studies with attribute authentication schemes could necessitate formal verification of these protocols.

**Data-origin authentication based on asymmetric keys:** This suggestion builds on PLA from [CLK05]. One idea could be to use the flow label field of IPv6 as public key in IDPKC. Instead of coupling the authenticity of an IP-packet with a specific entity, it is coupled with a key corresponding to a traffic class. Source authentication is achieved, as only an entity with the correct key would be able to send data signed with a given traffic class key. An E-node on the edge of a PCS could sign the IP-header with the correct private key for that service class. Intermediate E-nodes verify that the signature corresponds to the service class and forwards the packet along.

# References

[A$^+$03] William A Arbaugh et al. *Real 802.11 security: Wi-Fi protected access and 802.11 i.* Addison-Wesley Longman Publishing Co., Inc., 2003.

[ABB$^+$05] Alessandro Armando, David Basin, Yohan Boichut, Yannick Chevalier, Luca Compagna, Jorge Cuéllar, P Hankes Drielsma, Pierre-Cyrille Héam, Olga Kouchnarenko, Jacopo Mantovani, et al. The avispa tool for the automated validation of internet security protocols and applications. In *Computer Aided Verification*, pages 281–285. Springer, 2005.

[AE08] Bernard Aboba and Pasi Eronen. Extensible authentication protocol (eap) key management framework. Standards Track RFC 5247, Internet Engineering Task Force, 2008. https://tools.ietf.org/html/rfc5247, Accessed 19. May 2016.

[AJR11] Sufyan T Faraj Al-Janabi and Mayada Abdul-salam Rasheed. Public-key cryptography enabled kerberos authentication. In *Developments in E-systems Engineering (DeSE), 2011*, pages 209–214. IEEE, 2011.

[AN96] Martín Abadi and Roger Needham. Prudent engineering practice for cryptographic protocols. *IEEE transactions on Software Engineering*, 22(1):6, 1996.

[Ash97] Paul Ashley. Authorization for a large heterogeneous multi-domain system. In *Australian Unix and Open Systems Group National Conference*, pages 159–169, 1997.

[BBC$^+$98] Steven Blake, David Black, Mark Carlson, Elwyn Davies, Zheng Wang, and Walter Weiss. An architecture for differentiated services. 1998.

[BBKP07] Shane Balfe, Kent D Boklan, Zev Klagsbrun, and Kenneth G Paterson. Key refreshing in identity-based cryptography and its applications in manets. In *Military Communications Conference*, pages 1–8, 2007.

[BCM13] David Basin, Cas Cremers, and Simon Meier. Provably repairing the iso/iec 9798 standard for entity authentication. *Journal of Computer Security*, 21(6):817–846, 2013.

[BCS94] Robert Braden, David Clark, and Scott Shenker. Integrated services in the internet architecture: an overview. Technical report, 1994.

[BF01] Dan Boneh and Matt Franklin. Identity-based encryption from the weil pairing. In *Advances in Cryptology—CRYPTO 2001*, pages 213–229. Springer, 2001.

[BK12] Kashif Bashir and Mohammad Khalid Khan. Modification in kerberos assisted authentication in mobile ad-hoc networks to prevent ticket replay attacks. *International Journal of Engineering and Technology*, 4(3):307, 2012.

[Bla01] Bruno Blanchet. An efficient cryptographic protocol verifier based on prolog rules. In *csfw*, page 0082. IEEE, 2001.

[BLS04] Dan Boneh, Ben Lynn, and Hovav Shacham. Short signatures from the weil pairing. *Journal of Cryptology*, 17(4):297–319, 2004.

[BM13] Colin Boyd and Anish Mathuria. *Protocols for authentication and key establishment*. Springer Science & Business Media, 2013.

[BNN09] Mihir Bellare, Chanathip Namprempre, and Gregory Neven. Security proofs for identity-based identification and signature schemes. *Journal of Cryptology*, 22(1):1–61, 2009.

[CH14] Cas Cremers and Marko Horvat. Improving the iso/iec 11770 standard for key management techniques. In *Security Standardisation Research*, pages 215–235. Springer, 2014.

[CJW96] Neil Ching, Vicki Jones, and Marianne Winslett. Authorization in the digital library: Secure access to services across enterprise boundaries. In *Digital Libraries, 1996. ADL'96., Proceedings of the Third Forum on Research and Technology Advances in*, pages 110–119. IEEE, 1996.

[CKM00] David W Carman, Peter S Kruus, and Brian J Matt. Constraints and approaches for distributed sensor network security (final). *DARPA Project report,(Cryptographic Technologies Group, Trusted Information System, NAI Labs)*, 1(1), 2000.

[CLK05] Catharina Candolin, Janne Lundberg, and Hannu Kari. Packet level authentication in military networks. In *Proceedings of the 6th Australian Information Warfare & IT Security Conference*. Citeseer, 2005.

[CM10] Liqun Chen and Chris J Mitchell. Parsing ambiguities in authentication and key establishment protocols. *International Journal of Electronic Security and Digital Forensics*, 3(1):82–94, 2010.

[CNC$^{+}$04] Zhaohui Cheng, Manos Nistazakis, Richard Comley, White Hart Lane, London N Hr, et al. Id: A mandatory field in ike? Technical report, School of Computing Science, Middlesex University, 2004. http://www.wseas.us/e-library/conferences/athens2004/papers/487-418.pdf, Accessed 19.May 2016.

[Cre06] Casimier Joseph Franciscus Cremers. *Scyther: Semantics and verification of security protocols*. Eindhoven University of Technology, 2006.

[Cre08] Cas JF Cremers. The scyther tool: Verification, falsification, and analysis of security protocols. In *Computer aided verification*, pages 414–418. Springer, 2008.

[Dee98] Stephen E Deering. Internet protocol, version 6 (ipv6). Standards Track RFC 2460, Internet Engineering Task Force, 1998. https://tools.ietf.org/html/rfc2460, Accessed 19.May 2016.

[DH76] Whitfield Diffie and Martin E Hellman. New directions in cryptography. *Information Theory, IEEE Transactions on*, 22(6):644–654, 1976.

[Die08] Tim Dierks. The transport layer security (tls) protocol version 1.2. Standards Track RFC 5246, Internet Engineering Task Force, 2008. https://tools.ietf.org/html/rfc5246, Accessed 19.May 2016.

[Dif88] Whitfield Diffie. The first ten years of public-key cryptography. *Proceedings of the IEEE*, 76(5):560–577, 1988.

[DY83] Danny Dolev and Andrew C Yao. On the security of public key protocols. *Information Theory, IEEE Transactions on*, 29(2):198–208, 1983.

[EA04] Pasi Eronen and Jari Arkko. Authentication components: Engineering experiences and guidelines. In *Security Protocols*, pages 68–77. Springer, 2004.

[ES00] Carl Ellison and Bruce Schneier. Ten risks of pki: What you're not being told about public key infrastructure. *Comput Secur J*, 16(1):1–7, 2000.

[FALZ12] Victor Fajardo, Jari Arkko, John Loughney, and Glen Zorn. Diameter base protocol. Standards Track RFC 6733, Internet Engineering Task Force, 2012. https://tools.ietf.org/html/rfc6733, Accessed 19.May 2016.

[FD11] Anders Fongen and Norwegian Defence. Architecture patterns for a ubiquitous identity management system. *ICONS 2011*, pages 66–71, 2011.

[Fon10] Anders Fongen. Identity management without revocation. In *Emerging Security Information Systems and Technologies (SECURWARE), 2010 Fourth International Conference on*, pages 75–81. IEEE, 2010.

[FS86] Amos Fiat and Adi Shamir. How to prove yourself: Practical solutions to identification and signature problems. In *Advances in Cryptology—CRYPTO'86*, pages 186–194. Springer, 1986.

[GG09] David Galindo and Flavio D Garcia. A schnorr-like lightweight identity-based signature scheme. In *Progress in Cryptology–AFRICACRYPT 2009*, pages 135–148. Springer, 2009.

[GLSY04] Rosario Gennaro, Darren Leigh, Ravi Sundaram, and William Yerazunis. Batching schnorr identification scheme with applications to privacy-preserving authorization and low-bandwidth communication devices. In *Advances in Cryptology-ASIACRYPT 2004*, pages 276–292. Springer, 2004.

[GPS08] Steven D Galbraith, Kenneth G Paterson, and Nigel P Smart. Pairings for cryptographers. *Discrete Applied Mathematics*, 156(16):3113–3121, 2008.

[GT00] Joshua D Guttman and FJF Thayer. Protocol independence through disjoint encryption. In *Computer Security Foundations Workshop, 2000. CSFW-13. Proceedings. 13th IEEE*, pages 24–34. IEEE, 2000.

[HC$^+$98] Dan Harkins, Dave Carrel, et al. The internet key exchange (ike). Standards Track RFC 2409, Internet Engineering Task Force, 1998. https://tools.ietf.org/html/rfc2409, Accessed 19.May 2016.

[HG04] Katrin Hoeper and Guang Gong. Models of authentication in ad hoc networks and their related network properties. *International Association for Cryptologic Research*, 2004.

[HM01] Alan H Harbitter and Daniel A Menascé. Performance of public-key-enabled kerberos authentication in large networks. In *Security and Privacy, 2001. S&P 2001. Proceedings. 2001 IEEE Symposium on*, pages 170–183. IEEE, 2001.

[HO08] Geir Hallingstad and Sander Oudkerk. Protected core networking: an architectural approach to secure and flexible communications. *Communications Magazine, IEEE*, 46(11):35–41, 2008.

[HPSS08] Jeffrey Hoffstein, Jill Pipher, Joseph H Silverman, and Joseph H Silverman. *An introduction to mathematical cryptography*, volume 1. Springer, 2008.

[HTTm01] Matthew Hur, Brian Tung, Ryutov Tatyana, and more. Public key cryptography for cross-realm authentication in kerberos (pkcross). Internet draft, Internet Engineering Task Force, 2001. https://tools.ietf.org/html/draft-ietf-cat-kerberos-pk-cross-08. Accessed 24.May 2016.

[HW08] Anne Marie Hegland and Eli Winjum. Securing qos signaling in ip-based military ad hoc networks. *Communications Magazine, IEEE*, 46(11):42–48, 2008.

[HWH11] Anne Marie Hegland, Eli Winjum, and Ole-Erik Hedenstad. A framework for authentication in nbd tactical ad hoc networks. *Communications Magazine, IEEE*, 49(10):64–71, 2011.

[JN09] Marc Joye and Gregory Neven. *Identity-based cryptography*, volume 2. IOS press, 2009.

[Ken05] S Kent. Ip authentication header. Standards Track RFC 4302, Internet Engineering Task Force, 2005. https://tools.ietf.org/html/rfc4302, Accessed 19.May 2016.

[KH04] Kaoru Kurosawa and Swee-Huay Heng. From digital signature to id-based identification/signature. In *Public Key Cryptography–PKC 2004*, pages 248–261. Springer, 2004.

[KHNE14] C. Kaufman, P. Hoffman, Y. Nir, and P. Eronen. Internet key exchange protocol version 2 (ikev2). Standards Track RFC 7296, Internet Engineering Task Force, 2014. http://www.rfc-editor.org/info/rfc7296, Accessed 19. May 2016.

[KM05] Neal Koblitz and Alfred Menezes. Pairing-based cryptography at high security levels. In *Proceedings of Cryptography and Coding 2005, volume 3796 of LNCS*, 2005.

[KPP94] Per Kaijser, Tom Parker, and Denis Pinkas. Sesame: The solution to security for open distributed systems. *Computer Communications*, 17(7):501–518, 1994.

[Kra03] Hugo Krawczyk. Sigma: The 'sign-and-mac'approach to authenticated diffie-hellman and its use in the ike protocols. In *Advances in Cryptology-CRYPTO 2003*, pages 400–425. Springer, 2003.

[KS05] S Kent and K Seo. Security architecture for the internet protocol. Standards Track RFC 4301, Internet Engineering Task Force, 2005. https://tools.ietf.org/html/rfc4301, Accessed 19.May 2016.

[Len01] Arjen K Lenstra. Unbelievable security matching aes security using public key systems. In *Advances in Cryptology—ASIACRYPT 2001*, pages 67–86. Springer, 2001.

[Low96] Gavin Lowe. Breaking and fixing the needham-schroeder public-key protocol using fdr. In *Tools and Algorithms for the Construction and Analysis of Systems*, pages 147–166. Springer, 1996.

[Low97] Gavin Lowe. A hierarchy of authentication specifications. In *Computer Security Foundations Workshop, 1997. Proceedings., 10th*, pages 31–43. IEEE, 1997.

[Mei13] Simon Meier. Advancing automated security protocol verification. Phd thesis Nr. 20742, Eidgenössische Technische Hochschule ETH Zürich, 2013. http://e-collection.library.ethz.ch/eserv/eth:7011/eth-7011-02.pdf, Accessed 19.May 2016.

[MHN97] Ari Medvinsky, Matthew Hur, and C Neuman. Public key utilizing tickets for application servers (pktapp). Internet draft, Internet Engineering Task Force, 1997. https://shiraz.levkowetz.com/pdf/draft-ietf-cat-kerberos-pk-tapp-02.pdf. Accessed 24.May 2016.

[MMOB10] Shin'ichiro Matsuo, Kunihiko Miyazaki, Akira Otsuka, and David Basin. How to evaluate the security of real-life cryptographic protocols? In *Financial Cryptography and Data Security*, pages 182–194. Springer, 2010.

[MNSS88] S. P. Miller, B. C. Neuman, J. I. Schiller, and J. H. Saltzer. Kerberos authentication and authorization system. In *In Project Athena Technical Plan*, 1988.

[MPP$^+$15] Dustin Moody, Rene Peralta, Ray Perlner, Andrew Regenscheid, Allen Roginsky, and Lily Chen. Report on pairing-based cryptography. *Journal of Research of the National Institute of Standards and Technology*, 120:11, 2015.

[MSCB13] Simon Meier, Benedikt Schmidt, Cas Cremers, and David Basin. The tamarin prover for the symbolic analysis of security protocols. In *Computer Aided Verification*, pages 696–701. Springer, 2013.

[MY96] Ueli M Maurer and Yacov Yacobi. A non-interactive public-key distribution system. *Designs, Codes and Cryptography*, 9(3):305–316, 1996.

[NHYR05] Clifford Neuman, Sam Hartman, Tom Yu, and Kenneth Raeburn. The kerberos network authentication service (v5). Standards Track RFC 4120, Internet Engineering Task Force, 2005. https://www.ietf.org/rfc/rfc4120.txt, Accessed 19.May 2016.

[NS78] Roger M Needham and Michael D Schroeder. Using encryption for authentication in large networks of computers. *Communications of the ACM*, 21(12):993–999, 1978.

[PCN10] NATO RTO Task Group PCN. Requirements for a protected core networking (pcn) interoperability specification (ispec). *Research and Technology Organization, NATO UNCLASSIFIED RELEASABLE TO EAPC*, pages 1–211, 2010.

[PCN14] NATO RTO Task Group PCN. Selected challenges for protected core networking (pcn). *Research and Technology Organization, NATO UNCLASSIFIED RELEASABLE TO EAPC*, pages 1–46, 2014.

[PM04] Asad Amir Pirzada and Chris McDonald. Kerberos assisted authentication in mobile ad-hoc networks. In *Proceedings of the 27th Australasian conference on Computer science-Volume 26*, pages 41–46. Australian Computer Society, Inc., 2004.

[Pop02] Constantin Popescu. An efficient id-based group signature scheme. *Studia Univ. Babes-Bolyai, Informatica*, 47(2):29–38, 2002.

[RAJC11] Jarno Rajahalme, Shane Amante, Sheng Jiang, and Brian Carpenter. Ipv6 flow label specification. Standards Track RFC 6437, Internet Engineering Task Force, 2011. https://tools.ietf.org/html/rfc6437, Accessed 19. May 2016.

[RSA78] Ronald L Rivest, Adi Shamir, and Len Adleman. A method for obtaining digital signatures and public-key cryptosystems. *Communications of the ACM*, 21(2):120–126, 1978.

[SBW$^+$15] Mazda Salmanian, J David Brown, Susan Watson, Ronggong Song, Helen Tang, and Darcy Simmelink. An architecture for secure interoperability between coalition tactical manets. In *Military Communications Conference, MILCOM 2015-2015 IEEE*, pages 37–42. IEEE, 2015.

[SC97] Marvin A Sirbu and John Chung-I Chuang. Distributed authentication in kerberos using public key cryptography. In *Network and Distributed System Security, 1997. Proceedings., 1997 Symposium on*, pages 134–141. IEEE, 1997.

[SD03] Diana K Smetters and Glenn Durfee. Domain-based administration of identity-based cryptosystems for secure email and ipsec. In *USENIX Security*, 2003.

[Sha85] Adi Shamir. Identity-based cryptosystems and signature schemes. In *Advances in cryptology*, pages 47–53. Springer, 1985.

[SMD+13] Roland Schutz, Steve McLaughlin, Tim Daeleman, Marko Luoma, Markus Peuhkuri, Per Carlen, and John Haines. Protected core networking (pcn): Pcn qos and sla definition. In *2013 Military Communications and Information Systems Conference, MCC 2013, 7 October 2013 through 9 October 2013, Saint Malo, 1-8*, 2013.

[ST06] Shoichi Sakane and Michael Thomas. Kerberized internet negotiation of keys (kink). Standards Track RFC 4430, Internet Engineering Task Force, 2006. https://tools.ietf.org/html/rfc4430, Accessed 19.May 2016.

[Sta95] William Stallings. *Network and internetwork security: principles and practice*, volume 1. Prentice Hall Englewood Cliffs, 1995.

[Sti05] Douglas R Stinson. *Cryptography: theory and practice.* CRC press, 2005.

[SY16] S. Sorce and T. Yu. Kerberos authorization data container authenticated by multiple message authentication codes (macs). Standards Track RFC 7751, Internet Engineering Task Force, 2016. https://tools.ietf.org/html/rfc7751, Accessed 19.May 2016.

[TD07] Helen Tang and RD Defence. *Strong authentication for tactical mobile ad hoc networks.* Citeseer, 2007.

[Tha15] Sonia Thakur. A review on identity based cryptography. *International Journal of Computer Applications*, 119(13), 2015.

[Tsa08] Joe-Kai Tsay. Formal analysis of the kerberos authentication protocol. Phd thesis AAI3328667, University of Pennsylvania, 2008. http://www.lsv.ens-cachan.fr/Publis/PAPERS/PDF/these-JKT08.pdf, Accessed 19.May 2016.

[WRRS00] Steve Willens, Allan C Rubens, Carl Rigney, and William Allen Simpson. Remote authentication dial in user service (radius). Standards Track RFC 2865, Internet Engineering Task Force, 2000. https://tools.ietf.org/html/rfc2865. Accessed 19.May 2016.

[YO15] Huihui Yang and Vladimir A Oleshchuk. A dynamic attribute-based authentication scheme. In *Codes, Cryptology, and Information Security*, pages 106–118. Springer, 2015.

[ZT06] Larry Zhu and Brian Tung. Public key cryptography for initial authentication in kerberos (pkinit). Technical report, Available at http://tools.ietf.org/html/rfc4556.html. Access: 19.May 2016, 2006.

# Appendix A Internet Protocols

This appendix contains background information on the IPv6 standard. The flow label specification by IETF is [RAJC11]. The IPv6 specification itself is defined in [Dee98].

## A.1 Internet Protocol Security

Internet Protocol Security (IPSec) provides security services for traffic at the IP-layer [KS05]. Some of the services IPSec provides are access control, authentication, replay protection and integrity checks. Figure A.1 illustrates an IPv6 header. IPsec can enforce access control based on so called *selector* values in the IP-header. That is, determine if a given IP-packet can enter the network based on the value of certain fields. In order to do that properly, these fields must be linked with an identity so an authorization decision can be made. Integrity checks are necessary for the gateway to ensure that the packet was not modified after leaving the authorized sender. The Next Header Type in Figure A.1 can be used to indicate further headers, so called extension headers, which allow more intricate options for packet processing. One of these extension headers is Authentication Header (AH). An AH-header is appended to the packet if it is authenticated with IPSec AH. The extension header carries information regarding how this packet should be handled by the endpoint, or intermediate nodes.

### A.1.1 Authentication Header

IP Authentication Header is described in [Ken05]. The Authentication Header (AH) is an extension header to the IPv6 header to provide message authentication to an IP packet. Optionally, it can provide protection against replays. AH supports two modes of use, transport- and tunnel mode [Sta95]. Transport Mode authenticates IP payload and some of the IP header fields. Tunnel Mode authenticates the entire original IP packet and some of the outer IP-header fields. The rest of this section will be concerning the use of tunnel mode.

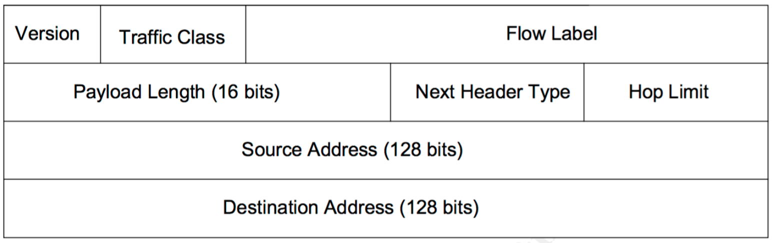

**Figure A.1:** Figure showing the fields of IPv6 header [Dee98].

IPv6 defines six extension headers, one of which is authentication header (AH). The AH provides integrity and data-origin authentication for the entire IPv6 packet. Table A.1 shows how the extension headers, and specifically the AH, are added to the original IPv6 packet in tunnel mode.

| | | | | | | |
|---|---|---|---|---|---|---|
| New IP HDR | Ext HDR* | AH | Orig IP HDR | Ext HDR* | TCP | Data |

**Table A.1:** A figure illustrating the structure of an IPv6 packet in Authentication Header tunnel mode [Ken05].

Figure A.2 illustrates the structure of Authentication Header. The security Parameters Index (SPI) is a number used to identify the security association. The sequence number field is a counter which starts at zero, and is incremented by one for each message exchanged in that security association. The goal of the sequence number is to prevent replay of old packets. Maybe the most interesting field is the authentication data field. It contains the Integrity Check Value (ICV) used to provide authentication and integrity to data. Given that the receiver possesses the secret key, it can compute a HMAC value over the received data in the same way as the sender and compare the output to the HMAC value received. Any changes of protected fields during transmission will lead to the receiver computing a different output value than the sender and the changes will be detected.

| 0 1 2 3 4 5 6 7 | 8 9 10 11 12 13 14 15 | 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 |
|---|---|---|
| Next Header | Payload Length | RESERVED |
| Security Parameter Index (SPI) | | |
| Sequence Number Field | | |
| Authentication Data (variable) | | |

**Figure A.2:** A figure showing the structure of the Authentication Header as specified in [Ken05]

### A.1.2 Establishment of Security Associations

Two methods of setting up security associations are discussed in this thesis. Internet Key Exchange and Kerberized Internet Negotiation of Keys (KINK). IKE is built on authenticated Diffie-Hellman key exchange, and KINK is built on Kerberos.

#### Internet Key Exchange

The sigma protocol family was proposed in [Kra03]. Sigma served as the cryptographic basis for signature based modes in IKEv1 [HC$^+$98], and a slightly modified version is used in IKEv2 [KHNE14]. Sigma, and therefore the IKE protocols, are based on a challenge-response mechanisms based on nonces for replay protection. Thus, IKE has a minimum of three messages passes. The IKEv2 protocol as shown in Figure A.3 consists of two message exchanges: IKE-SA-INIT and IKE-AUTH. The first exchange, IKE-SA-INIT set up a security association for the IKE protocol itself through agreement on cryptographic mechanisms and the exchange of material needed to create a shared key. In the second exchange, which is authenticated, information for the IPsec association is agreed upon.

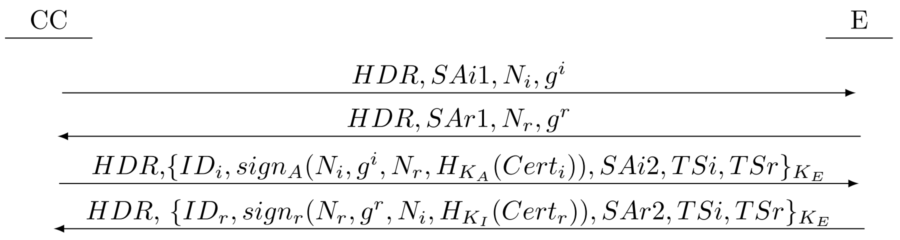
CC
E
$HDR, SAi1, N_i, g^i$
$HDR, SAr1, N_r, g^r$
$HDR, \{ID_i, sign_A(N_i, g^i, N_r, H_{K_A}(Cert_i)), SAi2, TSi, TSr\}_{K_E}$
$HDR, \{ID_r, sign_r(N_r, g^r, N_i, H_{K_I}(Cert_r)), SAr2, TSi, TSr\}_{K_E}$

**Figure A.3:** IKEv2 as standardized in [KHNE14]. Certificates are here used as an entity's identity.

Another way of setting up a security association is through Kerberized Internet Negotiation of Keys (KINK).

### KINK

The use of Kerberos [NHYR05] for setting up a security association for IPSec is described in IETF standard Request For Comments (RFC) 4430 KINK[ST06]. KINK reuses payloads from the Quick Mode of IKE version 1 [HC+98] in order to take advantage of existing implementations [ST06].

$$\begin{tikzcd}[column sep=huge] \text{CC} \arrow[r, shift left, "\text{CREATE+ISAKMP+KINK-AP-REQ}"] & \text{E} \arrow[l, shift left, "\text{REPLY+ISAKMP-KINK-AP-REP}"] \end{tikzcd}$$

**Figure A.4:** KINK as described in [ST06]. Through KINK, an IPSec SA can be set up in just two message passes. Kerberos is embedded in KINK as payload.

## A.2 Diameter

The diameter base protocol provides an AAA framework for services such as network access and QoS [FALZ12]. Information is transported within Attribute-Value Pairs (AVPs). Some AVPs are associated with the base protocol itself, while others are for specific services. The AVPs are transported within diameter messages between diameter clients and servers. In contrast to the Remote Authentication Dial-In User Service (RADIUS) protocol [WRRS00], any diameter-peer can initiate a request which makes its architecture mimic peer-to-peer networks. A diameter peer can act as client in one exchange, and server in another [FALZ12]. Typically, a client will be an entity at the boarder of the network performing access control. Figure A.5 illustrates this setup. The diameter client acts in many ways on behalf of the client, and can generate diameter authentication and authorization request messages. These requests are sent to a diameter server which provides these services. The diameter protocol specifies the encapsulation of EAP in AVPs, which in turn specify the encapsulation of entity authentication protocols.

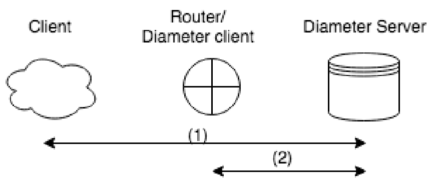

**Figure A.5:** In diameter the router acts as a diameter client, and forwards authentication traffic to a diameter server.

## A.3 Authorization within Kerberos

The information within this section is from RFC 7751 [SY16] and RFC 4120 [NHYR05]. Kerberos Authorization Data Container supersedes the previously used kdc-issued data element. It enables MACs or signatures to authenticate an authorization data container. The new authorization data container is called Container Authenticated by Multiple MACs (CAMMAC). The purpose of both kdc-issued and CAMMAC, is for the KDC to signal permissive usage of a ticket to the AS. This field is separately authenticated by the KDC so the application server has assurance that the decision to include these elements was made by the KDC. With the previous kdc-issued element, the field was authenticated by a key known by the KDC and the application server. The fact that the AS knows the key can cause a problem if an AS is using the ticket to ask the KDC for delegated authority. An AS would then be able to change values in the kdc-issued, and request delegated authority with increased permissions on behalf of the client. Using CAMMAC, a KDC could authenticate this field with a key known only by itself before issuing the ticket. It would then detect if an AS has changed the field later on.

```
AD-CAMMAC            ::=SEQUENCE {
    elements         [0] AuthorizationData,
    kdc-verifier     [1] Verifier-MAC OPTIONAL
    svc-verifier     [2] Verifier-MAC OPTIONAL
    other-verifiers  [3] SEQUENCE (SIZE (1..MAX))
                         OF Verifier OPTIONAL
}

Verifier             ::= CHOICE {
    mac              Verifier-MAC,
    ...
}
}
```

## A.4 PKDA

A Figure showing the PKDA protocol as described in [SC97]. In an attempt to distribute workload away from a central server, clients are in PKDA authenticating directly to Local Ticket Granting Servers (LTGSs) located with application servers.

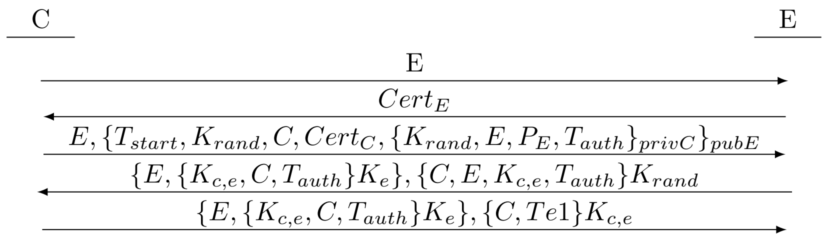
C
E
E
$Cert_E$
$E, \{T_{start}, K_{rand}, C, Cert_C, \{K_{rand}, E, P_E, T_{auth}\}_{privC}\}_{pubE}$
$\{E, \{K_{c,e}, C, T_{auth}\}K_e\}, \{C, E, K_{c,e}, T_{auth}\}K_{rand}$
$\{E, \{K_{c,e}, C, T_{auth}\}K_e\}, \{C, Te1\}K_{c,e}$

**Figure A.6:** PKDA as described in [SC97]. If instantiated with IDPKC, the first two messages are eliminated. The result is a three-pass protocol with timestamps.

# Appendix B Isabelle

The output from scyhter-proof can be further input to the Isabelle tool for independent theorem proving. To use Isabelle after installing scyther-proof as described in chapter 5, a few more supplementary tools must be present on the system. According to the Isabelle website http://isabelle.in.tum.de/installation.html some version of Perl 5.x with libwww is required (1,2,3). Then, download Isabelle from the website mentioned above, and unpack using (4):

```
(1) $ sudo apt-get install perlbrew
(2) $ perlbrew init
(3) $ perlbrew install perl-5-20.1
(4) $ tar -xzf Isabelle2016_app.tar.gz$
```

According the Meier's notes on the GitHub page[1], the logic image of Isabelle/HOL theories will be built the first time scyther-proof is called using the *isabelle* -flag.

[1]https://github.com/meiersi/scyther-proof

# Appendix C Code

This chapter contains code modelling the security protocols of chapter 4 and the security goals of chapter 5. It additionally contains the proof code output by scyther-proof after running the code against an attacker with Dolev-Yao capabilities.

## C.1 Protocol A

### C.1.1 Protocol model and goals

```
theory Kerberos_V5_4pass
begin

section{* 4-pass Kerberos Protocol, Version 5 *}

protocol KerberosFour
{

1_leak. C-> : L1
1. C -> A  : C,S,N1,L1, {'1',Author2, Author1, C, L1}sk(C)


2_leak. A -> : T1start, T1expire
2.   <- A: C, {'2_1', C, S, ServerKey, T1start, T1expire}k(A,S),
{'2_2', S, ServerKey, T1start, T1expire, N1,
{'2_2_1',Author3, A, C, T1start, T1expire}sk(A)}k(C,A)
C <-  : C, ServerTicket,
{'2_2', S, ServerKey, T1start, T1expire, N1,
{'2_2_1',Author3, A, C, T1start, T1expire}sk(A)}k(C,A)

3_leak. C -> : T1
3. C ->  : A, ServerTicket,
{'3', C, T1, {'3_1', Author4,
{'2_2_1', Author3, A, C, T1start, T1expire}sk(A), C, T1}sk(C)}ServerKey
       -> S: A, {'2_1', C, S, ServerKey, T1start, T1expire}k(A,S),
{'3', C, T1, {'3_1', Author4,
{'2_2_1', Author3, A, C, T1start, T1expire}sk(A), C, T1}sk(C)}ServerKey

4_leak. S -> : T2
4. S -> C : {'4', T2, {'4_1',Author5, S,C, T2}sk(S)}ServerKey

}

subsection{* Secrecy Properties *}

properties (of KerberosFour)
  A_SessionKey_secret: secret(A,-, SessionKey,{C,A})
  C_SessionKey_secret: secret(C,2, SessionKey,{C,A})
  A_ServerKey_secret: secret(A, −, ServerKey, {C,A,S})
  C_ServerKey_secret: secret(C, 2, ServerKey, {C,A,S})


subsection{* Authentication Properties *}

property (of KerberosFour) A_auth:
  premises
    "role(2) = A"
    "uncompromised(C#2, A#2, S#2)"
    "step(2, A_2)"
  imply a thread 1 such that
    "
     role(1) = C &
     C#2 = C#1 &
     Author1#2 = Author1#1 &
     Author2#2 = Author2#1 &
     L1#2 = L1#1 &

     St(2, A_1) < St(2, A_2)
    "

property (of KerberosFour) S_auth:
  premises
    "role(3) = S"
    "uncompromised(C#3, A#3, S#3)"
    "step(3, S_4)"
  imply threads 1, 2 such that
    "
     role(2) = A &
     C#3 = C#2 &
     A#3 = A#2 &
     S#3 = S#2 &
     T1start#3 = T1start#2 &
     T1expire#3 = T1expire#2 &
     ServerKey#3 = ServerKey#2 &
     Author3#3 = Author3#2 &

     role(1) = C &
     C#3  = C#1 &
     A#3  = A#1 &
     S#3  = S#1 &
     T1#3 = T1#1 &
     T1start#3 = T1start#1 &
     T1expire#3 = T1expire#1 &
     ServerKey#3 = ServerKey#1 &
     Author3#3 = Author3#1 &
     Author4#3 = Author4#1 &

     St(2,A_2) < St(1,C_2) <
     St(1,C_3) < St(3,S_3)
    "

property (of KerberosFour) C_auth:
  premises
    "role(1) = C"
    "uncompromised(C#1, A#1, S#1)"
    "step(1, C_4)"
  imply threads 2,3 such that
    "
     role(2) = A &
     A#1 = A#2 &
     C#1 = C#2 &
     T1start#1  = T1start#2 &
     T1expire#1 = T1expire#2 &
     Author3#1 = Author3#2 &

     role(3) = S &
     C#1  = C#3 &
     S#1  = S#3 &
```

```
      ServerKey#1 = ServerKey#3 &
      T1start#1 = T1start#3 &
      T1expire#1 = T1expire#3 &
      T2#1 = T2#3 &
      Author5#1 = Author5#3 &

      St(1, C_1) < St(2, A_1) <
      St(2, A_2) < St(1, C_2) <
      St(1, C_3) < St(3, S_3) <
      St(3, S_4) < St(1, C_4)
      "


end
```

### C.1.2 Proof script A

```
theory Kerberos_V5_4pass begin

section{* 4-pass Kerberos Protocol, Version 5 *}

protocol KerberosFour
{ role A
  { Recv_1(?C, ?S, ?N1, ?L1, {'1', ?Author2, ?Author1, ?C, ?L1}sk(?C))
    Send_2_leak(~T1start, ~T1expire)
    Send_2(?C, {'2_1', ?C, ?S, ~ServerKey, ~T1start, ~T1expire}k(A,?S),
            {'2_2', ?S, ~ServerKey, ~T1start, ~T1expire, ?N1,
             {'2_2_1', ~Author3, A, ?C, ~T1start, ~T1expire}sk(A)}
            k(?C,A)
           )
  }

  role C
  { Send_1_leak( ~L1 )
    Send_1(C, S, ~N1, ~L1, {'1', ~Author2, ~Author1, C, ~L1}sk(C))
    Recv_2(C, ?ServerTicket,
            {'2_2', S, ?ServerKey, ?T1start, ?T1expire, ~N1,
             {'2_2_1', ?Author3, ?A, C, ?T1start, ?T1expire}sk(?A)}
            k(C,?A)
           )
    Send_3_leak( ~T1 )
    Send_3(?A, ?ServerTicket,
         {'3', C, ~T1,
         {'3_1', ~Author4, {'2_2_1', ?Author3, ?A, C, ?T1start, ?T1expire}sk(?A),
         C, ~T1}
         sk(C)
           }
            ?ServerKey
           )
    Recv_4( {'4', ?T2, {'4_1', ?Author5, S, C, ?T2}sk(S)}?ServerKey )
  }

  role S
  { Recv_3(?A, {'2_1', ?C, S, ?ServerKey, ?T1start, ?T1expire}k(?A,S),
            {'3', ?C, ?T1,
             {'3_1', ?Author4,
              {'2_2_1', ?Author3, ?A, ?C, ?T1start, ?T1expire}sk(?A), ?C, ?T1}
             sk(?C)
            }
            ?ServerKey
           )
    Send_4_leak( ~T2 )
    Send_4( {'4', ~T2, {'4_1', ~Author5, S, ?C, ~T2}sk(S)}?ServerKey )
  }
}

property (of KerberosFour) KerberosFour_msc_typing:
  "A@C :: (Known(C_2) | Agent)
   A@S :: Known(S_3)
   Author1@A :: (Known(A_1) | Author1@C)
```

```
    Author2@A :: (Known(A_1) | Author2@C)
    Author3@C :: (Known(C_2) | Author3@A)
    Author3@S :: (Known(S_3) | Author3@A)
    Author4@S :: (Known(S_3) | Author4@C)
    Author5@C :: (Known(C_4) | Author5@S)
    C@A :: Known(A_1)
    C@S :: (Known(S_3) | Agent)
    L1@A :: Known(A_1)
    N1@A :: Known(A_1)
    S@A :: Known(A_1)
    ServerKey@C :: (Known(C_2) | ServerKey@A)
    ServerKey@S :: (Known(S_3) | ServerKey@A)
    ServerTicket@C :: Known(C_2)
    T1@S :: (Known(S_3) | T1@C)
    T1expire@C :: (Known(C_2) | T1expire@A)
    T1expire@S :: (Known(S_3) | T1expire@A)
    T1start@C :: (Known(C_2) | T1start@A)
    T1start@S :: (Known(S_3) | T1start@A)
    T2@C :: (Known(C_4) | T2@S)"
proof
  case A_1_Author1
  sources( {'1', ?Author2#0, ?Author1#0, ?C#0, ?L1#0}sk(?C#0) )
  qed (2 trivial)
next
  case A_1_Author2
  sources( {'1', ?Author2#0, ?Author1#0, ?C#0, ?L1#0}sk(?C#0) )
  qed (2 trivial)
next
  case A_1_C
  tautology
next
  case A_1_L1
  tautology
next
  case A_1_N1
  tautology
next
  case A_1_S
  tautology
next
  case C_2_A
  sources(
      {'2_2', S#0, ?ServerKey#0, ?T1start#0, ?T1expire#0, ~N1#0,
       {'2_2_1', ?Author3#0, ?A#0, C#0, ?T1start#0, ?T1expire#0}sk(?A#0)}
      k(C#0,?A#0) )
    case fake
    sources(
        {'2_2_1', ?Author3#0, ?A#0, C#0, ?T1start#0, ?T1expire#0}sk(?A#0) )
      case (C_3_enc_2 #1)
      sources(
          {'2_2', S#1, ?ServerKey#1, ?T1start#0, ?T1expire#0, ~N1#1,
           {'2_2_1', ?Author3#0, ?A#0, C#0, ?T1start#0, ?T1expire#0}sk(?A#0)}
          k(C#0,?A#0) )
      qed (2 trivial)
    qed (2 trivial)
  qed (1 trivial)
next
  case C_2_Author3
  sources(
      {'2_2', S#0, ?ServerKey#0, ?T1start#0, ?T1expire#0, ~N1#0,
       {'2_2_1', ?Author3#0, ?A#0, C#0, ?T1start#0, ?T1expire#0}sk(?A#0)}
      k(C#0,?A#0) )
    case fake
    sources(
        {'2_2_1', ?Author3#0, ?A#0, C#0, ?T1start#0, ?T1expire#0}sk(?A#0) )
      case (C_3_enc_2 #1)
      sources(
          {'2_2', S#1, ?ServerKey#1, ?T1start#0, ?T1expire#0, ~N1#1,
           {'2_2_1', ?Author3#0, ?A#0, C#0, ?T1start#0, ?T1expire#0}sk(?A#0)}
          k(C#0,?A#0) )
      qed (2 trivial)
    qed (2 trivial)
  qed (1 trivial)
next
  case C_2_ServerKey
  sources(
      {'2_2', S#0, ?ServerKey#0, ?T1start#0, ?T1expire#0, ~N1#0,
       {'2_2_1', ?Author3#0, ?A#0, C#0, ?T1start#0, ?T1expire#0}sk(?A#0)}
      k(C#0,?A#0) )
  qed (2 trivial)
next
  case C_2_ServerTicket
  tautology
next
  case C_2_T1expire
  sources(
      {'2_2', S#0, ?ServerKey#0, ?T1start#0, ?T1expire#0, ~N1#0,
       {'2_2_1', ?Author3#0, ?A#0, C#0, ?T1start#0, ?T1expire#0}sk(?A#0)}
      k(C#0,?A#0) )
  qed (2 trivial)
next
  case C_2_T1start
  sources(
      {'2_2', S#0, ?ServerKey#0, ?T1start#0, ?T1expire#0, ~N1#0,
       {'2_2_1', ?Author3#0, ?A#0, C#0, ?T1start#0, ?T1expire#0}sk(?A#0)}
      k(C#0,?A#0) )
  qed (2 trivial)
next
  case C_4_Author5
  sources(
      {'4', ?T2#0, {'4_1', ?Author5#0, S#0, C#0, ?T2#0}sk(S#0)}?ServerKey#0 )
    case fake
    sources( {'4_1', ?Author5#0, S#0, C#0, ?T2#0}sk(S#0) )
    qed (2 trivial)
  qed (1 trivial)
next
  case C_4_T2
  sources(
      {'4', ?T2#0, {'4_1', ?Author5#0, S#0, C#0, ?T2#0}sk(S#0)}?ServerKey#0 )
  qed (2 trivial)
next
  case S_3_A
  tautology
next
  case S_3_Author3
  sources(
      {'3', ?C#0, ?T1#0,
       {'3_1', ?Author4#0,
        {'2_2_1', ?Author3#0, ?A#0, ?C#0, ?T1start#0, ?T1expire#0}sk(?A#0),
        ?C#0, ?T1#0}
       sk(?C#0)
      }
      ?ServerKey#0 )
    case fake
    sources(
        {'3_1', ?Author4#0,
         {'2_2_1', ?Author3#0, ?A#0, ?C#0, ?T1start#0, ?T1expire#0}sk(?A#0),
         ?C#0, ?T1#0}
        sk(?C#0) )
      case fake
      sources(
          {'2_2_1', ?Author3#0, ?A#0, ?C#0, ?T1start#0, ?T1expire#0}sk(?A#0) )
        case (C_3_enc_2 #1)
        sources(
            {'2_2', S#1, ?ServerKey#1, ?T1start#0, ?T1expire#0, ~N1#1,
             {'2_2_1', ?Author3#0, ?A#0, C#1, ?T1start#0, ?T1expire#0}sk(?A#0)}
            k(C#1,?A#0) )
        qed (2 trivial)
      qed (2 trivial)
    next
      case (C_3_enc_1 #1)
      sources(
            {'2_2', S#1, ?ServerKey#1, ?T1start#0, ?T1expire#0, ~N1#1,
             {'2_2_1', ?Author3#0, ?A#0, C#1, ?T1start#0, ?T1expire#0}sk(?A#0)}
            k(C#1,?A#0) )
          case fake
          sources(
              {'2_2_1', ?Author3#0, ?A#0, C#1, ?T1start#0, ?T1expire#0}sk(?A#0) )
            case (C_3_enc_2 #2)
            sources(
                {'2_2', S#2, ?ServerKey#2, ?T1start#0, ?T1expire#0, ~N1#2,
                 {'2_2_1', ?Author3#0, ?A#0, C#1, ?T1start#0, ?T1expire#0}sk(?A#0)}
                k(C#1,?A#0) )
            qed (2 trivial)
          qed (2 trivial)
        qed (1 trivial)
      qed
    next
      case (C_3_enc #1)
      sources(
          {'2_2', S#1, ?ServerKey#0, ?T1start#0, ?T1expire#0, ~N1#1,
           {'2_2_1', ?Author3#0, ?A#0, C#1, ?T1start#0, ?T1expire#0}sk(?A#0)}
          k(C#1,?A#0) )
        case fake
        sources(
            {'2_2_1', ?Author3#0, ?A#0, C#1, ?T1start#0, ?T1expire#0}sk(?A#0) )
          case (C_3_enc_2 #2)
          sources(
              {'2_2', S#2, ?ServerKey#2, ?T1start#0, ?T1expire#0, ~N1#2,
               {'2_2_1', ?Author3#0, ?A#0, C#1, ?T1start#0, ?T1expire#0}sk(?A#0)}
              k(C#1,?A#0) )
          qed (2 trivial)
        qed (2 trivial)
      qed (1 trivial)
    qed
  next
    case S_3_Author4
    sources(
        {'3', ?C#0, ?T1#0,
         {'3_1', ?Author4#0,
          {'2_2_1', ?Author3#0, ?A#0, ?C#0, ?T1start#0, ?T1expire#0}sk(?A#0),
          ?C#0, ?T1#0}
         sk(?C#0)
        }
        ?ServerKey#0 )
      case fake
      sources(
          {'3_1', ?Author4#0,
           {'2_2_1', ?Author3#0, ?A#0, ?C#0, ?T1start#0, ?T1expire#0}sk(?A#0),
           ?C#0, ?T1#0}
          sk(?C#0) )
      qed (2 trivial)
    qed (1 trivial)
  next
    case S_3_C
    sources(
        {'2_1', ?C#0, S#0, ?ServerKey#0, ?T1start#0, ?T1expire#0}k(?A#0,S#0) )
    qed (2 trivial)
  next
    case S_3_ServerKey
    sources(
        {'2_1', ?C#0, S#0, ?ServerKey#0, ?T1start#0, ?T1expire#0}k(?A#0,S#0) )
    qed (2 trivial)
  next
    case S_3_T1
    sources(
        {'3', ?C#0, ?T1#0,
         {'3_1', ?Author4#0,
          {'2_2_1', ?Author3#0, ?A#0, ?C#0, ?T1start#0, ?T1expire#0}sk(?A#0),
          ?C#0, ?T1#0}
         sk(?C#0)
        }
        ?ServerKey#0 )
  qed (2 trivial)
next
  case S_3_T1expire
  sources(
      {'2_1', ?C#0, S#0, ?ServerKey#0, ?T1start#0, ?T1expire#0}k(?A#0,S#0) )
  qed (2 trivial)
next
  case S_3_T1start
  sources(
      {'2_1', ?C#0, S#0, ?ServerKey#0, ?T1start#0, ?T1expire#0}k(?A#0,S#0) )
  qed (2 trivial)
qed

subsection{* Secrecy Properties *}

property (of KerberosFour) A_SessionKey_secret:
  for all #0 the premises
    "role( #0 ) = A"
    "uncompromised( A#0, ?C#0 )"
    "knows(~SessionKey#0)"
  imply "False"
resolve 'KerberosFour_msc_typing
sources( ~SessionKey#0 )
qed

property (of KerberosFour) C_SessionKey_secret:
  for all #0 the premises
    "role( #0 ) = C"
    "uncompromised( C#0, ?A#0 )"
    "step(#0, C_2)"
    "knows(~SessionKey#0)"
  imply "False"
saturate
resolve 'KerberosFour_msc_typing
sources( ~SessionKey#0 )
qed

property (of KerberosFour) A_ServerKey_secret:
  for all #0 the premises
    "role( #0 ) = A"
    "uncompromised( A#0, ?C#0, ?S#0 )"
    "knows(~ServerKey#0)"
  imply "False"
resolve 'KerberosFour_msc_typing
sources( ~ServerKey#0 )
  case A_2_ServerKey
  contradicts secrecy of k(A#0,?S#0)
next
  case A_2_ServerKey_1
  contradicts secrecy of k(?C#0,A#0)
qed

property (of KerberosFour) C_ServerKey_secret:
  for all #0 the premises
    "role( #0 ) = C"
    "uncompromised( C#0, S#0, ?A#0 )"
    "step(#0, C_2)"
    "knows(?ServerKey#0)"
  imply "False"
saturate
resolve 'KerberosFour_msc_typing
sources(
    {'2_2', S#0, ?ServerKey#0, ?T1start#0, ?T1expire#0, ~N1#0,
     {'2_2_1', ?Author3#0, ?A#0, C#0, ?T1start#0, ?T1expire#0}sk(?A#0)}
    k(C#0,?A#0) )
  case fake
  contradicts secrecy of k(C#0,?A#0)
next
  case (A_2_enc_1 #1)
  contradictory due to 'A_ServerKey_secret'
qed
```

```
subsection{* Authentication Properties *}

property (of KerberosFour) A_auth:
  for all #2 the premises
    "role( #2 ) = A"
    "uncompromised( A#2, ?C#2, ?S#2 )"
    "step(#2, A_2)"
  imply
    "(? #1.
        role( #1 ) = C &
        ?C#2 = C#1 &
        ?Author1#2 = ~Author1#1 &
        ?Author2#2 = ~Author2#1 & ?L1#2 = ~L1#1 & St(#2, A_1) < St(#2, A_2))"
saturate
resolve 'KerberosFour_msc_typing
sources( {'1', ?Author2#2, ?Author1#2, ?C#2, ?L1#2}sk(?C#2) )
  case fake
  contradicts secrecy of sk(?C#2)
next
  case (C_1_enc #3)
  tautology
qed

property (of KerberosFour) S_auth:
  for all #3 the premises
    "role( #3 ) = S"
    "uncompromised( S#3, ?A#3, ?C#3 )"
    "step(#3, S_4)"
  imply
    "(? #1.
        (? #2.
           role( #2 ) = A &
           role( #1 ) = C &
           ?C#3 = ?C#2 &
           ?A#3 = A#2 &
           S#3 = ?S#2 &
           ?T1start#3 = ~T1start#2 &
           ?T1expire#3 = ~T1expire#2 &
           ?ServerKey#3 = ~ServerKey#2 &
           ?Author3#3 = ~Author3#2 &
           ?C#3 = C#1 &
           ?A#3 = ?A#1 &
           S#3 = S#1 &
           ?T1#3 = ~T1#1 &
           ?T1start#3 = ?T1start#1 &
           ?T1expire#3 = ?T1expire#1 &
           ?ServerKey#3 = ?ServerKey#1 &
           ?Author3#3 = ?Author3#1 &
           ?Author4#3 = ~Author4#1 &
           St(#2, A_2) < St(#1, C_2) &
           St(#1, C_2) < St(#1, C_3) & St(#1, C_3) < St(#3, S_3)))"
saturate
resolve 'KerberosFour_msc_typing
sources(
    {'2_1', ?C#3, S#3, ?ServerKey#3, ?T1start#3, ?T1expire#3}k(?A#3,S#3) )
  case fake
  contradicts secrecy of k(?A#3,S#3)
next
  case (A_2_enc #4)
  sources(
      {'3', ?C#3, ?T1#3,
       {'3_1', ?Author4#3,
        {'2_2_1', ?Author3#3, A#4, ?C#3, ~T1start#4, ~T1expire#4}sk(A#4), ?C#3,
        ?T1#3}
       sk(?C#3)
      }
      ~ServerKey#4 )
    case fake
    contradictory due to 'A_ServerKey_secret'
  next
```

```
    case (C_3_enc #5)
    sources(
        {'2_2', S#5, ~ServerKey#4, ~T1start#4, ~T1expire#4, ~N1#5,
         {'2_2_1', ?Author3#3, A#4, C#5, ~T1start#4, ~T1expire#4}sk(A#4)}
        k(C#5,A#4) )
      case fake
      contradicts secrecy of k(C#5,A#4)
    next
      case (A_2_enc_1 #6)
      tautology
    qed
  qed
qed

property (of KerberosFour) C_auth:
  for all #1 the premises
    "role( #1 ) = C"
    "uncompromised( C#1, S#1, ?A#1 )"
    "step(#1, C_4)"
  imply
    "(? #2.
        (? #3.
           role( #2 ) = A &
           role( #3 ) = S &
           ?A#1 = A#2 &
           C#1 = ?C#2 &
           ?T1start#1 = ~T1start#2 &
           ?T1expire#1 = ~T1expire#2 &
           ?Author3#1 = ~Author3#2 &
           C#1 = ?C#3 &
           S#1 = S#3 &
           ?ServerKey#1 = ?ServerKey#3 &
           ?T1start#1 = ?T1start#3 &
           ?T1expire#1 = ?T1expire#3 &
           ?T2#1 = ~T2#3 &
           ?Author5#1 = ~Author5#3 &
           St(#1, C_1) < St(#2, A_1) &
           St(#2, A_1) < St(#2, A_2) &
           St(#2, A_2) < St(#1, C_2) &
           St(#1, C_2) < St(#1, C_3) &
           St(#1, C_3) < St(#3, S_3) &
           St(#3, S_3) < St(#3, S_4) & St(#3, S_4) < St(#1, C_4)))"
saturate
resolve 'KerberosFour_msc_typing
sources(
    {'2_2', S#1, ?ServerKey#1, ?T1start#1, ?T1expire#1, ~N1#1,
     {'2_2_1', ?Author3#1, ?A#1, C#1, ?T1start#1, ?T1expire#1}sk(?A#1)}
    k(C#1,?A#1) )
  case fake
  contradicts secrecy of k(C#1,?A#1)
next
  case (A_2_enc_1 #2)
  sources( ~N1#1 )
    case C_1_N1
    sources(
        {'4', ?T2#1, {'4_1', ?Author5#1, S#1, C#1, ?T2#1}sk(S#1)}~ServerKey#2 )
      case fake
      contradictory due to 'A_ServerKey_secret'
    next
      case (S_4_enc #3)
      sources(
          {'3', C#1, ?T1#3,
           {'3_1', ?Author4#3,
            {'2_2_1', ?Author3#3, ?A#3, C#1, ?T1start#3, ?T1expire#3}sk(?A#3), C#1,
            ?T1#3}
           sk(C#1)
          }
          ~ServerKey#2 )
        case fake
        contradictory due to 'A_ServerKey_secret'
      next
```

```
        case (C_3_enc #4)
        sources(
            {'2_2', S#4, ~ServerKey#2, ?T1start#3, ?T1expire#3, ~N1#4,
             {'2_2_1', ?Author3#3, ?A#3, C#1, ?T1start#3, ?T1expire#3}sk(?A#3)}
            k(C#1,?A#3) )
          case fake
          contradictory due to 'A_ServerKey_secret'
        next
          case (A_2_enc_1 #5)
          tautology
        qed
      qed
    qed
  qed
qed

end
```

## C.2 Protocol B

Protocol model and security goals of the scenario where a CC authenticates using PKINIT Kerberos to a KDC is provided here. The corresponding proof code is in section C.2.2.

### C.2.1 Protocol B model and goals

```
theory PKINITKerberosNSL
begin

protocol PKINITKerberosNSL
{
1. C ->  : C,S, N1,L1,{'1_1',C,A, NC, Author1, Author2, L1}sk(C),
{'1_2',L1,N1, h(C,S, N1,L1,
{'1_1',C,A, NC, Author1, Author2, L1}sk(C))}sk(C)
     -> A: C,S, N1,L1,{'1_1',C,A, NC, Author1, Author2,L1}sk(C),
{'1_2',L1,N1, h(C,S, N1,L1,
{'1_1',C,A, NC, Author1, Author2, L1}sk(C))}sk(C)

2.   <- A: C, {'2_1', C, S, ServerKey, T1start,      T1expire}k(A,S),
{'2_2', S, ServerKey, T1start, T1expire, N1,
{'2_2_1', A,C, NC, Author3, T1start, T1expire}sk(A)}SessionKey,
{'2_3',{SessionKey, h(SessionKey, (C,S, N1,L1,
{'1_1',C,A, NC, Author1, Author2, L1}sk(C),
{'1_2', L1,N1, h(C,S, N1,L1,
{'1_1',C,A, NC, Author1, Author2,L1}sk(C))}sk(C)))}sk(A)}pk(C)
     C <-  : C, ServerTicket,
{'2_2', S, ServerKey, T1start, T1expire, N1,
{'2_2_1', A, C, NC, Author3, T1start, T1expire}sk(A)}SessionKey,
{'2_3',{SessionKey, h(SessionKey, (C,S, N1,L1,
{'1_1',C,A, NC, Author1, Author2, L1}sk(C),
{'1_2', L1,N1, h(C,S, N1,L1,
{'1_1',C,A,NC, Author1, Author2, L1}sk(C))}sk(C)))}sk(A)}pk(C)

3. C ->  : A, ServerTicket,
{'3', C, T1,{'3_1',C,S,NC2,AuthorX,Author4,T1}sk(C)}ServerKey
     -> S: A, {'2_1', C, S, ServerKey, T1start,  T1expire}k(A,S),
{'3', C, T1,{'3_1',C,S,NC2,AuthorX,Author4,T1}sk(C)}ServerKey
 4. S ->  : {'4', T2,{'4_1',S,C,NC2, Author5,T2}sk(S)}ServerKey
      -> C: {'4', T2,{'4_1',S,C,NC2, Author5,T2}sk(S)}ServerKey
}

properties (of PKINITKerberosNSL)
  A_SessionKey_secret: secret(A,-, SessionKey,{C,A})
  C_SessionKey_secret: secret(C,2, SessionKey,{C,A})
  A_ServerKey_secret: secret(A, −, ServerKey, {C,A,S})
  C_ServerKey_secret: secret(C, 2, ServerKey, {C,A,S})

property (of PKINITKerberosNSL) C_noninjective_synch:
  premises
    "role(1) = C"
    "step(1, C_2)"
    "uncompromised(C#1, A#1, S#1)"
  imply a thread 2 such that
    "  role(2) = A
     & A#1  = A#2
     & C#1  = C#2
     & NC#1 = NC#2
     & Author1#1 = Author1#2
     & Author2#1 = Author2#2
     & Author3#1 = Author3#2
     & L1#1 = L1#2
     & T1start#1 = T1start#2
     & T1expire#1 = T1expire#2
     & St(1, C_1) < St(2, A_1)
     & St(2, A_2) < St(1, C_2)
     & St(1, C_1) < St(1, C_2)
     & St(2, A_1) < St(2, A_2)
    "

property (of PKINITKerberosNSL) C2_noninjective_synch:
  premises
    "role(1) = C"
    "step(1,C_4)"
    "uncompromised(C#1, A#1, S#1)"
  imply a thread 3 such that
    "  role(3) = S
     & S#1 = S#3
     & C#1 = C#3
     & NC2#1 = NC2#3
     & T1#1 = T1#3
     & T2#1 = T2#3
     & AuthorX#1 = AuthorX#3
     & Author4#1 = Author4#3
     & Author5#1 = Author5#3

     & St(1, C_3) < St(3, S_3)
     & St(3, S_4) < St(1, C_4)
     & St(1, C_3) < St(1, C_4)
     & St(3, S_3) < St(3, S_4)
    "

property (of PKINITKerberosNSL) A_auth:
  premises
    "role(2) = A"
    "uncompromised(C#2, A#2, S#2)"
    "step(2, A_2)"
  imply a thread 1 such that
    "  role(1) = C
     & C#2 = C#1
     & NC#2 = NC#1
     & Author1#2 = Author1#1
     & Author2#2 = Author2#1
     & N1#2 = N1#1
     & L1#2 = L1#1

     & St(1, C_1) < St(2, A_1)
     & St(2, A_1) < St(2, A_2)
    "
property (of PKINITKerberosNSL) S_auth:
  premises
    "role(3) = S"
    "uncompromised(C#3, A#3, S#3)"
    "step(3, S_4)"
  imply threads 1, 2 such that
    "
```

```
      role(2) = A &
      C#3 = C#2 &
      A#3 = A#2 &
      S#3 = S#2 &
      T1start#3 = T1start#2 &
      T1expire#3 = T1expire#2 &
      ServerKey#3 = ServerKey#2 &

      role(1) = C &
      C#3  = C#1 &
      A#3  = A#1 &
      S#3  = S#1 &
      NC2#3 = NC2#1 &
      T1#3 = T1#1 &
      AuthorX#3 = AuthorX#1 &
      Author4#3 = Author4#1 &
      T1start#3 = T1start#1 &
      T1expire#3 = T1expire#1 &
      ServerKey#3 = ServerKey#1 &

      St(1,C_1) < St(2,A_1) <
      St(2,A_2) < St(1,C_2) <
      St(1,C_3) < St(3,S_3) <
      St(3,S_4)
    "

property (of PKINITKerberosNSL) C_auth:
  premises
    "role(1) = C"
    "uncompromised(C#1, A#1, S#1)"
    "step(1, C_4)"
  imply threads 2, 3 such that
    "
      role(2) = A &
      A#1 = A#2 &
      C#1 = C#2 &
      NC#1 = NC#2 &
      SessionKey#1  = SessionKey#2 &
      T1start#1  = T1start#2 &
      T1expire#1 = T1expire#2 &

      role(3) = S &
      C#1  = C#3 &
      S#1  = S#3 &
      NC2#1 = NC2#3 &
      ServerKey#1 = ServerKey#3 &
      T2#1 = T2#3 &

      St(1, C_1) < St(2, A_1) <
      St(2, A_2) < St(1, C_2) <
      St(1, C_3) < St(3, S_3) <
      St(3, S_4) < St(1, C_4)
      "


end
```

### C.2.2 Proof script B

```
theory PKINIT_Kerberos_V5_WithNSL begin

protocol PKINITKerberosNSL
{ role A
  { Recv_1(?C, ?S, ?N1, ?L1,
            {'1_1', ?C, A, ?NC, ?Author1, ?Author2, ?L1}sk(?C),
            {'1_2', ?L1, ?N1,
             h(?C, ?S, ?N1, ?L1, {'1_1', ?C, A, ?NC, ?Author1, ?Author2, ?L1}sk(?C))}
            sk(?C)
          )
    Send_2_leak(~T1start, ~T1expire)
    Send_2(?C, {'2_1', ?C, ?S, ~ServerKey, ~T1start, ~T1expire}k(A,?S),
    {'2_2', ?S, ~ServerKey, ~T1start, ~T1expire, ?N1,
    {'2_2_1', A, ?C, ?NC, ~Author3, ~T1start, ~T1expire}sk(A)}
           ~SessionKey,
           {'2_3',
            {~SessionKey,
        h(~SessionKey, ?C, ?S, ?N1, ?L1,
            {'1_1', ?C, A, ?NC, ?Author1, ?Author2, ?L1}sk(?C),
            {'1_2', ?L1, ?N1,
            h(?C, ?S, ?N1, ?L1, {'1_1', ?C, A, ?NC, ?Author1, ?Author2, ?L1}sk(?C))}
            sk(?C)
             )
            }
            sk(A)
           }
           pk(?C)
          )
  }

  role C
  { Send_1_leak( ~L1 )
    Send_1(C, S, ~N1, ~L1, {'1_1', C, A, ~NC, ~Author1, ~Author2, ~L1}sk(C),
           {'1_2', ~L1, ~N1,
            h(C, S, ~N1, ~L1, {'1_1', C, A, ~NC, ~Author1, ~Author2, ~L1}sk(C))}
           sk(C)
          )
    Recv_2(C, ?ServerTicket,
           {'2_2', S, ?ServerKey, ?T1start, ?T1expire, ~N1,
            {'2_2_1', A, C, ~NC, ?Author3, ?T1start, ?T1expire}sk(A)}
           ?SessionKey,
           {'2_3',
            {?SessionKey,
             h(?SessionKey, C, S, ~N1, ~L1,
                 {'1_1', C, A, ~NC, ~Author1, ~Author2, ~L1}sk(C),
                 {'1_2', ~L1, ~N1,
                  h(C, S, ~N1, ~L1, {'1_1', C, A, ~NC, ~Author1, ~Author2, ~L1}sk(C))}
                 sk(C)
               )
            }
            sk(A)
           }
           pk(C)
          )
    Send_3_leak( ~T1 )
    Send_3(A, ?ServerTicket,
           {'3', C, ~T1, {'3_1', C, S, ~NC2, ~AuthorX, ~Author4, ~T1}sk(C)}
           ?ServerKey
          )
    Recv_4( {'4', ?T2, {'4_1', S, C, ~NC2, ?Author5, ?T2}sk(S)}?ServerKey )
  }

  role S
  { Recv_3(?A, {'2_1', ?C, S, ?ServerKey, ?T1start, ?T1expire}k(?A,S),
           {'3', ?C, ?T1, {'3_1', ?C, S, ?NC2, ?AuthorX, ?Author4, ?T1}sk(?C)}
           ?ServerKey
          )
    Send_4_leak( ~T2 )
    Send_4( {'4', ~T2, {'4_1', S, ?C, ?NC2, ~Author5, ~T2}sk(S)}?ServerKey )
  }
}

property (of PKINITKerberosNSL) PKINITKerberosNSL_msc_typing:
  "A@S :: Known(S_3)
   Author1@A :: (Known(A_1) | Author1@C)
   Author2@A :: (Known(A_1) | Author2@C)
   Author3@C :: (Known(C_2) | Author3@A)
   Author4@S :: (Known(S_3) | Author4@C)
   Author5@C :: (Known(C_4) | Author5@S)
   AuthorX@S :: (Known(S_3) | AuthorX@C)
   C@A :: Known(A_1)
   C@S :: (Known(S_3) | Agent)
```

```
     L1@A :: Known(A_1)
     N1@A :: Known(A_1)
     NC@A :: (Known(A_1) | NC@C)
     NC2@S :: (Known(S_3) | NC2@C)
     S@A :: Known(A_1)
     ServerKey@C :: (Known(C_2) | ServerKey@A)
     ServerKey@S :: (Known(S_3) | ServerKey@A)
     ServerTicket@C :: Known(C_2)
     SessionKey@C :: (Known(C_2) | SessionKey@A)
     T1@S :: (Known(S_3) | T1@C)
     T1expire@C :: (Known(C_2) | T1expire@A)
     T1expire@S :: (Known(S_3) | T1expire@A)
     T1start@C :: (Known(C_2) | T1start@A)
     T1start@S :: (Known(S_3) | T1start@A)
     T2@C :: (Known(C_4) | T2@S)"
proof
  case A_1_Author1
  sources(
      {'1_1', ?C#0, A#0, ?NC#0, ?Author1#0, ?Author2#0, ?L1#0}sk(?C#0) )
  qed (2 trivial)
next
  case A_1_Author2
  sources(
      {'1_1', ?C#0, A#0, ?NC#0, ?Author1#0, ?Author2#0, ?L1#0}sk(?C#0) )
  qed (2 trivial)
next
  case A_1_C
  tautology
next
  case A_1_L1
  tautology
next
  case A_1_N1
  tautology
next
  case A_1_NC
  sources(
      {'1_1', ?C#0, A#0, ?NC#0, ?Author1#0, ?Author2#0, ?L1#0}sk(?C#0) )
  qed (2 trivial)
next
  case A_1_S
  tautology
next
  case C_2_Author3
  sources(
      {'2_2', S#0, ?ServerKey#0, ?T1start#0, ?T1expire#0, ~N1#0,
       {'2_2_1', A#0, C#0, ~NC#0, ?Author3#0, ?T1start#0, ?T1expire#0}sk(A#0)}
      ?SessionKey#0 )
    case fake
    sources(
        {'2_2_1', A#0, C#0, ~NC#0, ?Author3#0, ?T1start#0, ?T1expire#0}sk(A#0) )
    qed (2 trivial)
  qed (1 trivial)
next
  case C_2_ServerKey
  sources(
      {'2_2', S#0, ?ServerKey#0, ?T1start#0, ?T1expire#0, ~N1#0,
       {'2_2_1', A#0, C#0, ~NC#0, ?Author3#0, ?T1start#0, ?T1expire#0}sk(A#0)}
      ?SessionKey#0 )
  qed (2 trivial)
next
  case C_2_ServerTicket
  tautology
next
  case C_2_SessionKey
  sources(
      {'2_2', S#0, ?ServerKey#0, ?T1start#0, ?T1expire#0, ~N1#0,
       {'2_2_1', A#0, C#0, ~NC#0, ?Author3#0, ?T1start#0, ?T1expire#0}sk(A#0)}
      ?SessionKey#0 )
  qed (2 trivial)
next
```

```
  case C_2_T1expire
  sources(
      {'2_2', S#0, ?ServerKey#0, ?T1start#0, ?T1expire#0, ~N1#0,
       {'2_2_1', A#0, C#0, ~NC#0, ?Author3#0, ?T1start#0, ?T1expire#0}sk(A#0)}
      ?SessionKey#0 )
  qed (2 trivial)
next
  case C_2_T1start
  sources(
      {'2_2', S#0, ?ServerKey#0, ?T1start#0, ?T1expire#0, ~N1#0,
       {'2_2_1', A#0, C#0, ~NC#0, ?Author3#0, ?T1start#0, ?T1expire#0}sk(A#0)}
      ?SessionKey#0 )
  qed (2 trivial)
next
  case C_4_Author5
  sources(
      {'4', ?T2#0, {'4_1', S#0, C#0, ~NC2#0, ?Author5#0, ?T2#0}sk(S#0)}
      ?ServerKey#0 )
    case fake
    sources( {'4_1', S#0, C#0, ~NC2#0, ?Author5#0, ?T2#0}sk(S#0) )
    qed (2 trivial)
  qed (1 trivial)
next
  case C_4_T2
  sources(
      {'4', ?T2#0, {'4_1', S#0, C#0, ~NC2#0, ?Author5#0, ?T2#0}sk(S#0)}
      ?ServerKey#0 )
  qed (2 trivial)
next
  case S_3_A
  tautology
next
  case S_3_Author4
  sources(
      {'3', ?C#0, ?T1#0,
       {'3_1', ?C#0, S#0, ?NC2#0, ?AuthorX#0, ?Author4#0, ?T1#0}sk(?C#0)}
      ?ServerKey#0 )
    case fake
    sources(
        {'3_1', ?C#0, S#0, ?NC2#0, ?AuthorX#0, ?Author4#0, ?T1#0}sk(?C#0) )
    qed (2 trivial)
  qed (1 trivial)
next
  case S_3_AuthorX
  sources(
      {'3', ?C#0, ?T1#0,
       {'3_1', ?C#0, S#0, ?NC2#0, ?AuthorX#0, ?Author4#0, ?T1#0}sk(?C#0)}
      ?ServerKey#0 )
    case fake
    sources(
        {'3_1', ?C#0, S#0, ?NC2#0, ?AuthorX#0, ?Author4#0, ?T1#0}sk(?C#0) )
    qed (2 trivial)
  qed (1 trivial)
next
  case S_3_C
  sources(
      {'2_1', ?C#0, S#0, ?ServerKey#0, ?T1start#0, ?T1expire#0}k(?A#0,S#0) )
  qed (2 trivial)
next
  case S_3_NC2
  sources(
      {'3', ?C#0, ?T1#0,
       {'3_1', ?C#0, S#0, ?NC2#0, ?AuthorX#0, ?Author4#0, ?T1#0}sk(?C#0)}
      ?ServerKey#0 )
    case fake
    sources(
        {'3_1', ?C#0, S#0, ?NC2#0, ?AuthorX#0, ?Author4#0, ?T1#0}sk(?C#0) )
    qed (2 trivial)
  qed (1 trivial)
next
  case S_3_ServerKey
  sources(
      {'2_1', ?C#0, S#0, ?ServerKey#0, ?T1start#0, ?T1expire#0}k(?A#0,S#0) )
  qed (2 trivial)
next
  case S_3_T1
  sources(
      {'3', ?C#0, ?T1#0,
       {'3_1', ?C#0, S#0, ?NC2#0, ?AuthorX#0, ?Author4#0, ?T1#0}sk(?C#0)}
      ?ServerKey#0 )
  qed (2 trivial)
next
  case S_3_T1expire
  sources(
      {'2_1', ?C#0, S#0, ?ServerKey#0, ?T1start#0, ?T1expire#0}k(?A#0,S#0) )
  qed (2 trivial)
next
  case S_3_T1start
  sources(
      {'2_1', ?C#0, S#0, ?ServerKey#0, ?T1start#0, ?T1expire#0}k(?A#0,S#0) )
  qed (2 trivial)
qed

subsection{* Secrecy Properties *}

property (of PKINITKerberosNSL) A_SessionKey_secret:
  for all #0 the premises
    "role( #0 ) = A"
    "uncompromised( A#0, ?C#0 )"
    "knows(~SessionKey#0)"
  imply "False"
resolve 'PKINITKerberosNSL_msc_typing
sources( ~SessionKey#0 )
  case A_2_SessionKey
  contradicts secrecy of sk(?C#0)
qed

property (of PKINITKerberosNSL) C_SessionKey_secret:
  for all #0 the premises
    "role( #0 ) = C"
    "uncompromised( A#0, C#0 )"
    "step(#0, C_2)"
    "knows(?SessionKey#0)"
  imply "False"
saturate
resolve 'PKINITKerberosNSL_msc_typing
sources(
    {'2_2', S#0, ?ServerKey#0, ?T1start#0, ?T1expire#0, ~N1#0,
     {'2_2_1', A#0, C#0, ~NC#0, ?Author3#0, ?T1start#0, ?T1expire#0}sk(A#0)}
    ?SessionKey#0 )
  case fake
  sources(
      {'2_2_1', A#0, C#0, ~NC#0, ?Author3#0, ?T1start#0, ?T1expire#0}sk(A#0) )
    case fake
    contradicts secrecy of sk(A#0)
  next
    case (A_2_enc_2 #1)
    contradictory due to 'A_SessionKey_secret'
  qed
next
  case (A_2_enc_1 #1)
  contradictory due to 'A_SessionKey_secret'
qed

property (of PKINITKerberosNSL) A_ServerKey_secret:
  for all #0 the premises
    "role( #0 ) = A"
    "uncompromised( A#0, ?C#0, ?S#0 )"
    "knows(~ServerKey#0)"
  imply "False"
resolve 'PKINITKerberosNSL_msc_typing
sources( ~ServerKey#0 )
```

```
  case A_2_ServerKey
  contradicts secrecy of k(A#0,?S#0)
next
  case A_2_ServerKey_1
  contradictory due to 'A_SessionKey_secret'
qed

property (of PKINITKerberosNSL) C_ServerKey_secret:
  for all #0 the premises
    "role( #0 ) = C"
    "uncompromised( A#0, C#0, S#0 )"
    "step(#0, C_2)"
    "knows(?ServerKey#0)"
  imply "False"
saturate
resolve 'PKINITKerberosNSL_msc_typing
sources(
    {'2_2', S#0, ?ServerKey#0, ?T1start#0, ?T1expire#0, ~N1#0,
     {'2_2_1', A#0, C#0, ~NC#0, ?Author3#0, ?T1start#0, ?T1expire#0}sk(A#0)}
    ?SessionKey#0 )
  case fake
  contradictory due to 'C_SessionKey_secret'
next
  case (A_2_enc_1 #1)
  contradictory due to 'A_ServerKey_secret'
qed

subsection{* Authentication Properties *}

property (of PKINITKerberosNSL) C_noninjective_synch:
  for all #1 the premises
    "role( #1 ) = C"
    "uncompromised( A#1, C#1, S#1 )"
    "step(#1, C_2)"
  imply
    "(? #2.
        role( #2 ) = A &
        A#1 = A#2 &
        C#1 = ?C#2 &
        ~NC#1 = ?NC#2 &
        ~Author1#1 = ?Author1#2 &
        ~Author2#1 = ?Author2#2 &
        ?Author3#1 = ~Author3#2 &
        ~L1#1 = ?L1#2 &
        ?T1start#1 = ~T1start#2 &
        ?T1expire#1 = ~T1expire#2 &
        St(#1, C_1) < St(#2, A_1) &
        St(#2, A_2) < St(#1, C_2) &
        St(#1, C_1) < St(#1, C_2) & St(#2, A_1) < St(#2, A_2))"
saturate
resolve 'PKINITKerberosNSL_msc_typing
sources(
    {'2_2', S#1, ?ServerKey#1, ?T1start#1, ?T1expire#1, ~N1#1,
     {'2_2_1', A#1, C#1, ~NC#1, ?Author3#1, ?T1start#1, ?T1expire#1}sk(A#1)}
    ?SessionKey#1 )
  case fake
  contradictory due to 'C_SessionKey_secret'
next
  case (A_2_enc_1 #2)
  sources( {'1_1', C#1, A#1, ~NC#1, ?Author1#2, ?Author2#2, ?L1#2}sk(C#1) )
    case fake
    contradicts secrecy of sk(C#1)
  next
    case (C_1_enc #3)
    tautology
  qed
qed

property (of PKINITKerberosNSL) C2_noninjective_synch:
  for all #1 the premises
    "role( #1 ) = C"
    "uncompromised( A#1, C#1, S#1 )"
    "step(#1, C_4)"
  imply
    "(? #3.
        role( #3 ) = S &
        S#1 = S#3 &
        C#1 = ?C#3 &
        ~NC2#1 = ?NC2#3 &
        ~T1#1 = ?T1#3 &
        ?T2#1 = ~T2#3 &
        ~AuthorX#1 = ?AuthorX#3 &
        ~Author4#1 = ?Author4#3 &
        ?Author5#1 = ~Author5#3 &
        St(#1, C_3) < St(#3, S_3) &
        St(#3, S_4) < St(#1, C_4) &
        St(#1, C_3) < St(#1, C_4) & St(#3, S_3) < St(#3, S_4))"
saturate
resolve 'PKINITKerberosNSL_msc_typing
sources(
    {'4', ?T2#1, {'4_1', S#1, C#1, ~NC2#1, ?Author5#1, ?T2#1}sk(S#1)}
    ?ServerKey#1 )
  case fake
  contradictory due to 'C_ServerKey_secret'
next
  case (S_4_enc #2)
  sources(
      {'3', C#1, ?T1#2,
       {'3_1', C#1, S#1, ~NC2#1, ?AuthorX#2, ?Author4#2, ?T1#2}sk(C#1)}
      ?ServerKey#1 )
    case fake
    contradictory due to 'C_ServerKey_secret'
  next
    case (C_3_enc #3)
    tautology
  qed
qed

property (of PKINITKerberosNSL) A_auth:
  for all #2 the premises
    "role( #2 ) = A"
    "uncompromised( A#2, ?C#2, ?S#2 )"
    "step(#2, A_2)"
  imply
    "(? #1.
        role( #1 ) = C &
        ?C#2 = C#1 &
        ?NC#2 = ~NC#1 &
        ?Author1#2 = ~Author1#1 &
        ?Author2#2 = ~Author2#1 &
        ?N1#2 = ~N1#1 &
        ?L1#2 = ~L1#1 & St(#1, C_1) < St(#2, A_1) & St(#2, A_1) < St(#2, A_2))"
saturate
resolve 'PKINITKerberosNSL_msc_typing
sources(
    {'1_2', ?L1#2, ?N1#2,
     h(?C#2, ?S#2, ?N1#2, ?L1#2,
       {'1_1', ?C#2, A#2, ?NC#2, ?Author1#2, ?Author2#2, ?L1#2}sk(?C#2))
    }
    sk(?C#2) )
  case fake
  contradicts secrecy of sk(?C#2)
next
  case (C_1_enc_1 #3)
  tautology
qed

property (of PKINITKerberosNSL) S_auth:
  for all #3 the premises
    "role( #3 ) = S"
    "uncompromised( S#3, ?A#3, ?C#3 )"
    "step(#3, S_4)"
  imply
    "(? #1.
        (? #2.
           role( #2 ) = A &
           role( #1 ) = C &
           ?C#3 = ?C#2 &
           ?A#3 = A#2 &
           S#3 = ?S#2 &
           ?T1start#3 = ~T1start#2 &
           ?T1expire#3 = ~T1expire#2 &
           ?ServerKey#3 = ~ServerKey#2 &
           ?C#3 = C#1 &
           ?A#3 = A#1 &
           S#3 = S#1 &
           ?NC2#3 = ~NC2#1 &
           ?T1#3 = ~T1#1 &
           ?AuthorX#3 = ~AuthorX#1 &
           ?Author4#3 = ~Author4#1 &
           ?T1start#3 = ?T1start#1 &
           ?T1expire#3 = ?T1expire#1 &
           ?ServerKey#3 = ?ServerKey#1 &
           St(#1, C_1) < St(#2, A_1) &
           St(#2, A_1) < St(#2, A_2) &
           St(#2, A_2) < St(#1, C_2) &
           St(#1, C_2) < St(#1, C_3) &
           St(#1, C_3) < St(#3, S_3) & St(#3, S_3) < St(#3, S_4)))"
saturate
resolve 'PKINITKerberosNSL_msc_typing
sources(
    {'2_1', ?C#3, S#3, ?ServerKey#3, ?T1start#3, ?T1expire#3}k(?A#3,S#3) )
  case fake
  contradicts secrecy of k(?A#3,S#3)
next
  case (A_2_enc #4)
  sources(
      {'1_1', ?C#3, A#4, ?NC#4, ?Author1#4, ?Author2#4, ?L1#4}sk(?C#3) )
    case fake
    contradicts secrecy of sk(?C#3)
  next
    case (C_1_enc #5)
    sources(
        {'3', C#5, ?T1#3,
         {'3_1', C#5, S#3, ?NC2#3, ?AuthorX#3, ?Author4#3, ?T1#3}sk(C#5)}
        ~ServerKey#4 )
      case fake
      contradictory due to 'A_ServerKey_secret'
    next
      case (C_3_enc #6)
      sources(
          {'2_2', S#3, ~ServerKey#4, ?T1start#6, ?T1expire#6, ~N1#6,
           {'2_2_1', A#6, C#5, ~NC#6, ?Author3#6, ?T1start#6, ?T1expire#6}sk(A#6)}
          ?SessionKey#6 )
        case fake
        contradictory due to 'A_ServerKey_secret'
      next
        case (A_2_enc_1 #7)
        tautology
      qed
    qed
  qed
qed

property (of PKINITKerberosNSL) C_auth:
  for all #1 the premises
    "role( #1 ) = C"
    "uncompromised( A#1, C#1, S#1 )"
    "step(#1, C_4)"
  imply
    "(? #2.
        (? #3.
           role( #2 ) = A &
           role( #3 ) = S &
           A#1 = A#2 &
           C#1 = ?C#2 &
           ~NC#1 = ?NC#2 &
           ?SessionKey#1 = ~SessionKey#2 &
           ?T1start#1 = ~T1start#2 &
           ?T1expire#1 = ~T1expire#2 &
           C#1 = ?C#3 &
           S#1 = S#3 &
           ~NC2#1 = ?NC2#3 &
           ?ServerKey#1 = ?ServerKey#3 &
           ?T2#1 = ~T2#3 &
           St(#1, C_1) < St(#2, A_1) &
           St(#2, A_1) < St(#2, A_2) &
           St(#2, A_2) < St(#1, C_2) &
           St(#1, C_2) < St(#1, C_3) &
           St(#1, C_3) < St(#3, S_3) &
           St(#3, S_3) < St(#3, S_4) & St(#3, S_4) < St(#1, C_4)))"
saturate
resolve 'PKINITKerberosNSL_msc_typing
sources(
    {'2_2', S#1, ?ServerKey#1, ?T1start#1, ?T1expire#1, ~N1#1,
     {'2_2_1', A#1, C#1, ~NC#1, ?Author3#1, ?T1start#1, ?T1expire#1}sk(A#1)}
    ?SessionKey#1 )
  case fake
  contradictory due to 'C_SessionKey_secret'
next
  case (A_2_enc_1 #2)
  sources( {'1_1', C#1, A#1, ~NC#1, ?Author1#2, ?Author2#2, ?L1#2}sk(C#1) )
    case fake
    contradicts secrecy of sk(C#1)
  next
    case (C_1_enc #3)
    sources(
        {'4', ?T2#1, {'4_1', S#1, C#1, ~NC2#1, ?Author5#1, ?T2#1}sk(S#1)}
        ~ServerKey#2 )
      case fake
      contradictory due to 'A_ServerKey_secret'
    next
      case (S_4_enc #3)
      sources(
          {'3', C#1, ?T1#3,
           {'3_1', C#1, S#1, ~NC2#1, ?AuthorX#3, ?Author4#3, ?T1#3}sk(C#1)}
          ~ServerKey#2 )
        case fake
        contradictory due to 'A_ServerKey_secret'
      next
        case (C_3_enc #4)
        tautology
      qed
    qed
  qed
qed

end
```