Project Number	AC095
Project Title	ASPeCT: Advanced Security for Personal Communications Technologies
Document Type	Major Deliverable
Security Class	Public

Deliverable Number	D19
Title of Deliverable	Report on final trial and demonstration
Nature of Deliverable	Report
Document Reference	AC095/PFN/W12/DS/P/19/1
Contributing Work Packages	WP2.1; WP2.2; WP2.3; WP2.4; WP2.5.
Contractual Date of Delivery	March 98 (Y04M01)
Actual Date of Delivery	April 98 (Y04M02)
Editors	Eleni Papadopoulou, Panafon S.A. section 2.1 : Geneviève Vanneste, Siemens ATEA section 2.2 : Keith Howker, Royal Holloway University of London section 2.3 : Herman Verrelst, Katholieke Universiteit Leuven

Abstract	This deliverable describes the demonstrations of the use of the UIM with access to end-to-end security services and billing security services, as well as trials of the fraud management tool and the migration performance.

Keywords	ACTS ASPeCT authentication billing certificates cryptography electronic commerce EXODUS fraud detection GSM integrity micropayment	migration neural network non-repudiation rule-based security SIM smart card trial TTP value-added services UIM UMTS

TABLE OF CONTENTS

TABLE OF FIGURES

TABLE OF TABLES

0. Introduction

0.1 Executive Summary

This document is a Major Deliverable that contains input from all the technical workpackages which are involved in the project trials. All the scheduled trials and demonstrations that reflect the final stage of the project work are described, with the exception of the work of WP2.7 on biometric authentication, which is the subject of separate deliverables. The security solutions for the third generation of mobile communications that were developed within ASPeCT are introduced. The corresponding implementations for the trials are presented in detail. Methods for collecting meaningful information from the trials, by both quantitative and qualitative indicators, as well as for evaluating the results are provided.

The work areas whose solutions will be presented and evaluated through trials include authentication in UMTS with multi-application smart cards, secure billing applications with Trusted Third Party services, and fraud detection.

The structure of this deliverable is formed around the three trials that will be conducted within the current - and final - project year. The first Section is an overview of the trials, providing condensed information on the major ASPeCT events. The second Section contains the details of the planned trials and demonstrations. Both Sections are divided in three sub-sections corresponding to the three main areas of ASPeCT work.

Every trial is covered with respect to various aspects, each forming a different chapter in the second Section.

The Technical Description chapter contains the background of the security solutions that is largely independent of the trial implementation.

The Trial Configuration and Methodology chapter contains the specific trial description covering all organisational and procedural aspects for performing the trials, collecting and evaluating the results.

The Demonstration Configuration chapter describes the additional or different functionalities that can be tested in a demonstration environment.

The Realisation chapter includes the software details of the specific trial implementations.

Finally, the Plan chapter contains the milestones for these events as well as the dependencies on third parties and any potential risks involved in consequence.

0.2 Document Control

0.2.1 Document History

Version	Date	
Draft A01	16-Feb-98	First skeleton draft to solicit input
Draft A	04-Mar-98	First draft
Draft B	23-Mar-98	Second draft
Draft C	15-Apr-98	Final draft
1	30-Apr-98	Final document

0.3 Abbreviations and Acronyms

ACRYL	Advanced Cryptographic Library (by Siemens)
ACTS	Advanced Communications Technologies and Services
AMV	Agnew-Mullin-Vanstone (equation)
APDU	Application Protocol Data Unit
ASN.1	Abstract Syntax Notation (version 1)
ASPeCT	Advanced Security for Personal Communications Technologies
ATM	Asynchronous Transfer Mode
ATT	Attributed Toll Tickets
AuC	Authentication Centre
BALM	B-number analysis ALarM
BRUTUS	B-number and RUle-based analysis of Toll tickets utilizing Unsupervised and Supervised neural network technologies
CA	Certification Authority
CBC	Cipher Block Chaining mode
COLA	COnversion LAyer
CoLa	Conversion Layer
CPN	Calling Party Number
CRL	Certificate Revocation List
CSA	Certificate Service Application
CUP	Current User Profile
CUSF	Call Unrelated Service Function
DECT	Digital Enhanced Cordless Telecommunications
DES	Data Encryption Standard
DIB	Directory Information Base
DIT	Directory Information Tree
DLL	Dynamic Link Library
EEPROM	Electrically Erasable Programmable Read Only Memory
EMV	Europay, Mastercard and Visa
ETSI	European Telecommunications Standards Institute
EXODUS	EXperiments On the Deployment of UMTS
FP	Fixed Part (DECT)
FPAI	Fixed Point of Attachment Identifier
FSM	Finite State Machine

GSM	Global System for Mobile communications
GUI	Graphical User Interface
HTML	HyperText Markup Language
HTTP	HyperText Transfer Protocol
HW	Hardware
IEC	International Electrotechnical Commission
IMSI	International Mobile Subscriber Identity
IMUI	International Mobile User Identity
IMUN	International Mobile User Number
IN	Intelligent Network
INAP	Intelligent Network Application Protocol
IPF	Internal Public key File
ISF	Internal Secret key File
ISO	International Standards Organisation
IV	Initialization Vector
IWU	Interworking unit
LAI	Local Area Identifier
MAC	Message Authentication Code
MB	Mobile Broadband
NN	Neural Network
NNI	Network Node Interface
NO	Network Operator
OFB	Output FeedBack mode
PABX	Private Automatic Branch Exchange
PDAT	Protocol Data Analysis Tool
PDU	Protocol Data Unit
PIN	Personal Identification Number
PP	Portable Part (DECT)
RACE	Research and development in Advanced Communication technologies in Europe
RIPEMD	RACE Integrity Primitives Evaluation Message Digest
RNG	Random Number Generator
ROM	Read Only Memory
RSA	Rivest, Shamir and Adleman (public key algorithm)
SALR	Supervised NN ALaRm
SALV	Supervised NN ALarm leVel
SCF	Service Control Function

SCP	Service Control Point
SIM	Subscriber Identity Module
SLP	Service Logic Program
SMG	Special Mobile Group
SP	Service Provider
SPI	Service Provider Identity
SSP	Service Switching Point
SW	Software
TBNB	Tag for B-NumBer
TBTP	Tag for B TyPe of number
TBZC	Toll ticket B-number Zone Code
TCDR	Tag for Chargeable DuRation
TCP/IP	Transmission Control Protocol / Internet Protocol
TCSD	Tag for Charge Start Date
TCST	Tag for Charge Start Time
TMUI	Temporary Mobile User Identity
TSDN	Toll ticket Starting Date Normalised
TSTS	Toll ticket Starting Time in Seconds
TT	Toll Ticket
TTP	Trusted Third Party
UALM	Unsupervised NN ALarM
UIM	User Identity Module
UMTS	Universal Mobile Telecommunications System
UNI	User Network Interface
UPH	User Profile History
UPR	User Profile Record
USIM	UMTS SIM
UTC	Universal Time Coordinates
VAS	Value Added Service
VASP	Value Added Service Provider
VCI	Virtual Channel Identifier
VPI	Virtual Path Identifier
Winsocks	Windows Sockets
WP	Work Package
WWW	World Wide Web

0.4 Notation

Cryptographic Notation The notation used in Section 2.2 for cryptographic processes is based on the convention given in ISO/IEC 9798-1 [ISO98].

0.5 Document Cross References

[ANSI81] ANSI X3.92, American National Standard for Data Encryption Algorithm (DEA), 1981

[ANSI83] ANSI X3.106, American National Standard for Information Systems - Data Encryption Algorithm - Modes of Operation, 1983

[ASP/EX] ASPeCT/EXODUS Annex to Memorandum of Understanding on Joint ASPeCT-EXODUS Trials, 16-NOV-97

[BP95] A Bosselaers, B Preneel, (Eds). Integrity Primitives for Secure Information Systems, Final Report of RACE Integrity Primitives Evaluation (RIPE - RACE R1040), LNCS 1007, Springer-Verlag, 1995

[D02] ACTS AC095, ASPeCT Deliverable D02, Initial report on security requirements

Ref. AC095/ATEA/W21/DS/P/02/1

[D05] ACTS AC095, ASPeCT Deliverable D05, Migration Scenarios

Ref. AC095/ATEA/W21/DS/P/05/1

[D06] ACTS AC095, ASPeCT Deliverable D06, Definition of Fraud Detection Concepts

Ref. AC095/KUL/W22/DS/P/06/1

[D07] ACTS AC095, ASPeCT Deliverable D07, Security Services: First Specification

Ref. AC095/RHUL/W23/DS/I/07/1

[D08] ACTS AC095, ASPeCT Deliverable D08, Fraud Management Tools: First Prototype

Ref. AC095/RHUL/W22/DS/P/08/1

[D09] ACTS AC095, ASPeCT Deliverable D09, Trusted Third Parties: First implementation

Ref. AC095/RHUL/W23/DS/P/09/1

[D10] ACTS AC095, ASPeCT Deliverable D10, Secure billing: First implementation

Ref. AC095/SAG/W25/DS/P/10/1

[D11] ACTS AC095, ASPeCT Deliverable D11, Limiting Smart Card Constraints

Ref. AC095/GD/W24/DS/P/11/1

[D13] ACTS AC095, ASPeCT Deliverable D13, Fraud Management Tools: Evaluation Report

Ref. AC095/SAG/W22/DS/P/13/1

[D18] ACTS AC095, ASPeCT Deliverable D18, Fraud Detection Concepts: Final Report

Ref. AC095/VOD/W22/DS/P/18/1

[Fle87] Fletcher R. "Practical methods of optimization" 2nd ed. Chichester and New York: John Wiley and Sons, 1987

[Gra91] Grabec I (1991). Modelling of Chaos by a Self-Organizing Neural Network. Artificial Neural Networks. Proceedings of ICANN, Espoo, Vol 1, eds. Kohonen T, Mäkisara K, Simula O, Kangas J. pp 151-156, Elsevier Science Publications

[ISO95] ISO/IEC 7816-4: 1995 - Identification cards - Integrated circuit(s) cards with contacts - Part 4: Interindustry commands for interchange

[ISO96a] ISO/IEC 2nd CD 14888-3. Information technology - Security techniques - Digital signature with appendix - Part 3: Certificate-based mechanisms, June 1996

[ISO96b] ISO/IEC 9796-2 (review). Information technology - Security techniques - Digital signature techniques giving message recovery - Part 2: Mechanisms using a hash function, June 1996

[ISO98] ISO/IEC 9798-1 Information technology - Security techniques - Entity authentication

[Koh88] Kohonen T (1988). Self-organization and Associative Memory, Springer Verlag, Berlin, pp119

[Ped95] Pedersen T P, Electronic payments of small amounts, DAIMI PB-495, Computer Science Department, Aarhus University, August 1995

[PrEN96] *pr*EN1546-2, 1996-01-23 (Committee Draft).

"Identification card systems – Intersector Electronic Purse – Part 2: Security Architecture"

[V100] ASPeCT/ DOC/VOD/100/WP2.1/C, Timings and Measurements for the Trial

1. Trial and Demonstration Overview

1.1 Summary of Authentication Trial

1.1.1 Background

The need for enhanced security features in UMTS, has led to the definition of specific security objectives within ETSI SMG10. These objectives have been translated into security requirements, resulting in a classification of security features. Both secret key-based and public key-based mechanisms have been proposed for UMTS to provide mutual authentication, cipher key agreement for confidentiality, anonymity and non-repudiation.

1.1.2 ASPeCT results on authentication

The work done on UMTS authentication, as part of ASPeCT, has led to the following technical results.

- To enable migration from GSM to UMTS, a multi-application card has been defined and implemented, containing a GSM SIM application and a preliminary UMTS USIM application, offering the security mechanisms proposed in ASPeCT for user authentication.
- To achieve flexible introduction of new authentication mechanisms and algorithms, a framework for authentication has been introduced, with the ability to negotiate which authentication protocol shall be used.
- In order to facilitate roaming in a network with a large number of Network Operators and Service Providers, the need has been identified for roaming agreements to be set-up dynamically, as and when they are required.

1.1.3 The authentication framework

The framework, in combination with a public key authentication mechanism, has been integrated in an experimental UMTS environment provided by the EXODUS project.

The authentication framework allows the authentication capabilities of USIMs, Network Operators (NOs) and Service Providers (SPs) to be taken into consideration when selecting which mechanism to use. A list of capability classes (including the mechanisms supported) is maintained so that different entities (USIMs, NOs, SPs) can negotiate the mechanisms to be used.

The framework can be split into two phases: the *negotiation phase* and the *user-network authentication phase.* The negotiation phase ends with the agreement of a mechanism to be used in the user-network authentication, and may include the setting up of a roaming agreement, where one has not previously existed.

In Figure 1.1.1, the operational scenario is shown for a new user roaming in a network, with a roaming agreement already existing between the Network Operator and the Service Provider.

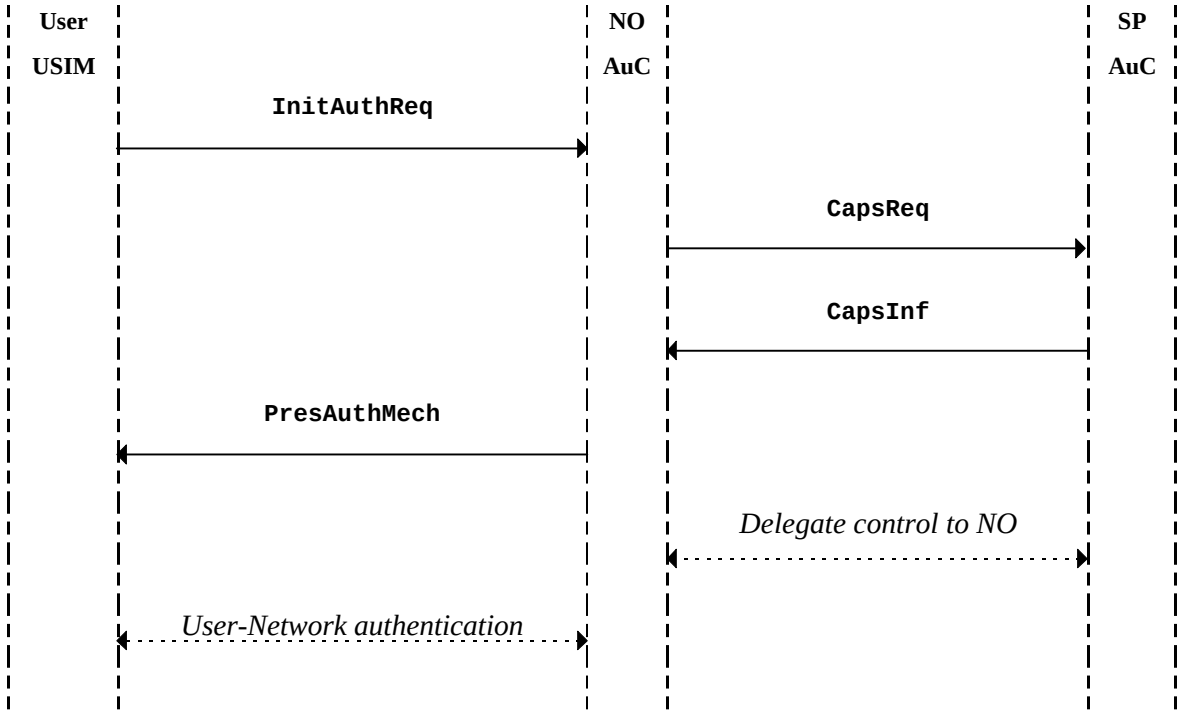

Figure 1.1.1 - The User is not registered and a roaming agreement exists

The UMTS authentication trial uses an ASPeCT mutual authentication protocol between smart cards (or USIMs) attached to EXODUS terminals, and an ASPeCT AuC attached to an EXODUS SCP. In the trial, ASPeCT security services are integrated into the EXODUS signalling system. The experimental UMTS platform provided by EXODUS is enhanced by the authentication functionality provided by ASPeCT. The main objective of the trial is to show the feasibility of the implementation on a real network.

Communication between the ASPeCT terminal and the network is provided by DECT and fixed broadband access. The terminal itself is enhanced with the necessary ASPeCT software to interface with a smart card-based USIM connected to the terminal. The user part of the authentication protocol runs on the ASPeCT USIM.

The UMTS network functionality is provided by the EXODUS core network. The security mechanisms are incorporated into the EXODUS core network using an ASPeCT AuC connected to the EXODUS SCP. Authentication is conducted over INAP, using the `Authentication-req` operation. Interrogation of the Service Provider is realised using the INAP `HandleInformationRequest` operation.

The trial configuration is shown in Figure 1.1.2. This configuration is available across two UMTS Islands in Basel and Milan. Within this configuration the distinction between the Network Operator and the Service Provider can not be made. Each SCP has home subscribers and visiting subscribers. The AuC connected to the SCP will fulfil two roles, home-AuC and visited-AuC.

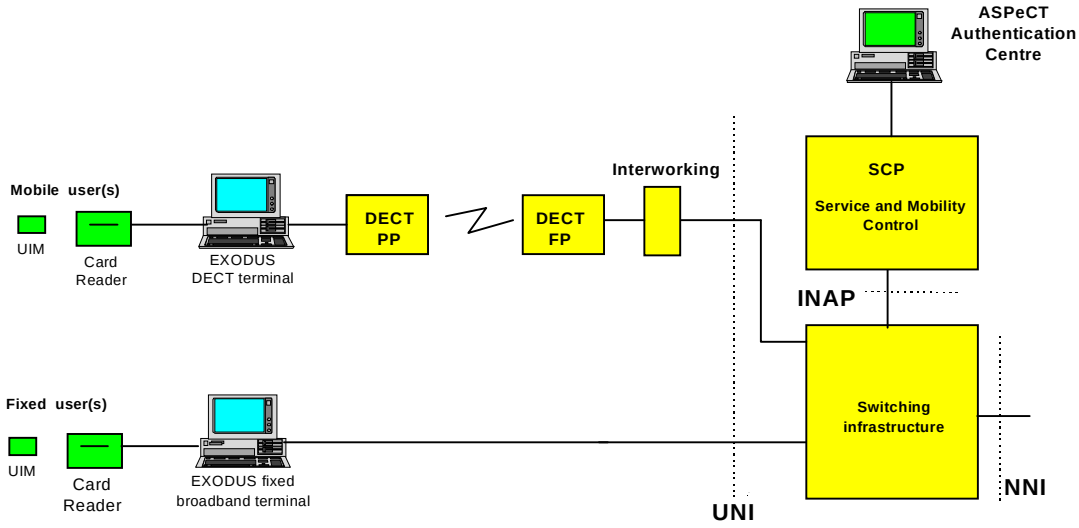

Figure 1.1.2 - Trial Configuration Of The Basel Island

The EXODUS users participating in the Multimedia Trial and the Healthcare Trial will be using authentication when using the main mobility management services or by accessing IN services.

Within ASPeCT some dedicated authentication trials will be done, evaluating the performance of the different possible authentication scenarios.

1.2 Summary of Secure Billing and TTP Trial

1.2.1 Overview

The results of the ASPeCT work on trusted third parties and secure billing are presented in four steps:

- two first demonstrators (demo 1);
- one joint second demonstrator (demo 2);
- trials in which the ASPeCT second demonstrator will be integrated with the EXODUS platform;
- an enhanced demonstrator, with greater functionality, which will be conducted subsequently to this trial.

Two separate first demonstrators for trusted third party (TTP) services and for secure billing services were completed in February 1997 [D09], [D10].

The first demonstrators will be combined in a joint TTP and secure billing second demonstrator in Quarter 1, 1998. The purpose of combining the first demonstrators is to show that the entities involved in secure billing can make on-line use of TTP services.

The second demonstrator will be integrated with the EXODUS platform to form the joint ASPeCT-EXODUS 'TTP and secure billing field trial' in accordance with our Memorandum of Understanding [ASP/EX]. The purpose of the field trial is to show that the proposed security mechanisms work in an experimental UMTS environment, and to measure and analyze the technical feasibility of the system in absolute terms and the acceptability of system performance and quality of service as perceived by users.

1.2.2 Secure Billing

UMTS will only be a success if the mobile user is able to choose from a much larger variety of services than those offered by today's networks, if he (the user) can be billed for their use, and where the billing information is accurate and is incontestable by all parties. The ASPeCT secure billing work concentrates on a new secure scheme to bill users for Value-Added Services (VASs), which provide value-added information to the user.

It is expected that UMTS users will possess terminals with much larger processing and display capabilities than today's mainly speech-orientated terminals. Personal mobile communicators will integrate the functions of a mobile telephone and a laptop or palmtop PC. These devices may be used to access more advanced information services than those available to users of VASs in mobile systems today. For instance, instead of being restricted to character-orientated displays, the user will be able to display hypertext documents which include graphics. In the trial, the user will be able to retrieve hypertext documents from a VAS provider.

The charging scheme for value-added services must be secure against cheating, and the parties involved must have the assurance that justified claims relating to charges can be proved, and that unjustified claims cannot be made successfully. In addition, the scheme must be flexible and efficient in terms of the processing power required at the user side and bandwidth required, because a radio interface might be involved. ASPeCT has demonstrated a proposed charging scheme for VASs in UMTS which satisfies the above requirements. The scheme is a credit-based payment scheme using micro-payments.

The security features implemented in the trial also include end-to-end mutual authentication between the user and the VAS Provider (VASP), which does not involve the UMTS Network Operator or Service Provider.

For the first demonstrator it was assumed that the necessary security information (in particular, public key certificates) had been obtained by the user and VASP in advance. If this is the case, then the only on-line communication required in the charging procedure is between the user and the VASP; the TTP need not be on-line.

For the second demonstrator however, it is assumed that the certificates have yet to be retrieved (and checked against a revocation list) from a Trusted Third Party, which is on-line during the charging procedure.

The VAS will be realized by an HTTP client application on the mobile user side and an HTTP server application on the VASP side. Both applications will communicate end-to-end over a data channel set up between the mobile user and the VASP.

1.2.3 Trusted Third Parties

A well established role for trusted third parties is the generation, distribution and management of public key certificates. Such certification services will become increasingly important in future mobile telecommunication services as public key-based security services become more widespread. In the trial the ASPeCT TTP provides on-line certification services to the mobile user and the VASP, who require certificates as part of the secure billing service. The certificate services here include time-stamping and signature of the certificates provided to the VASP, together with some basic certificate management (off-line) operations.

1.2.4 Configuration of trial

The TTP and secure billing field trial involves three entities: a mobile user, a VASP and a TTP, each represented by an EXODUS terminal. The entities each have fixed broadband access to the EXODUS experimental UMTS platform.

The configuration for the trial is shown below.

The trial session will consist of three steps:

1. set up data connections: user to VASP, and VASP to TTP;
2. launch TTP and secure billing applications;
3. access value-added information service using TTP and secure billing services.

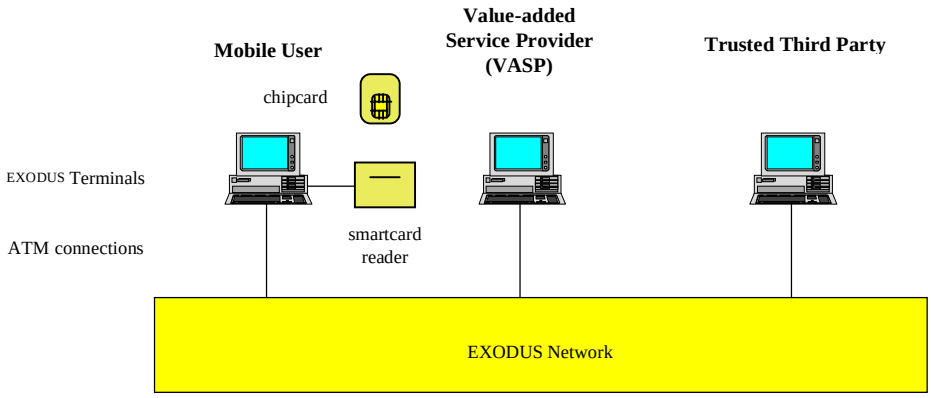

Figure 1.2.1 - Trial configuration

1.3 Summary of Fraud Detection Trial

The efforts of ASPeCT on fraud detection have focused on finding algorithms and software tools for detecting fraud in mobile networks. Three different approaches have been studied: rule-based systems, supervised neural networks and unsupervised neural networks. The first demonstration, where the three original tools were tested, showed the validity of the basic fraud detection concepts and compared them. The results were analyzed with respect to the identified requirements, to the defined functionality and to the aimed quality of service. The outcome of this analysis was fed back to the demonstrators to enhance their functionality. Each method's strengths and weaknesses were thus perceived. Also, it was made clearer that the interconnection of the tools to form one enhanced integrated system could combine all components' strong points.

The integrated detection tool is a serial implementation of the separate systems. This serial implementation - shown in Figure 1.3.1 - will use the Unsupervised Neural Network and B-number analysis tool to generate additional information for detecting fraudulent behavior, but will pass this information to the Supervised Neural Network for initial analysis. This will then raise any further indications of fraud that it detects, and pass all of this information on to the Rule Based tool. Once this has processed all the data, a monitoring tool will store and display any raised alarms. Individual modules forward information that they receive from any other module and add tagged information of their own should it be required. It is this integrated system that is used within the trials.

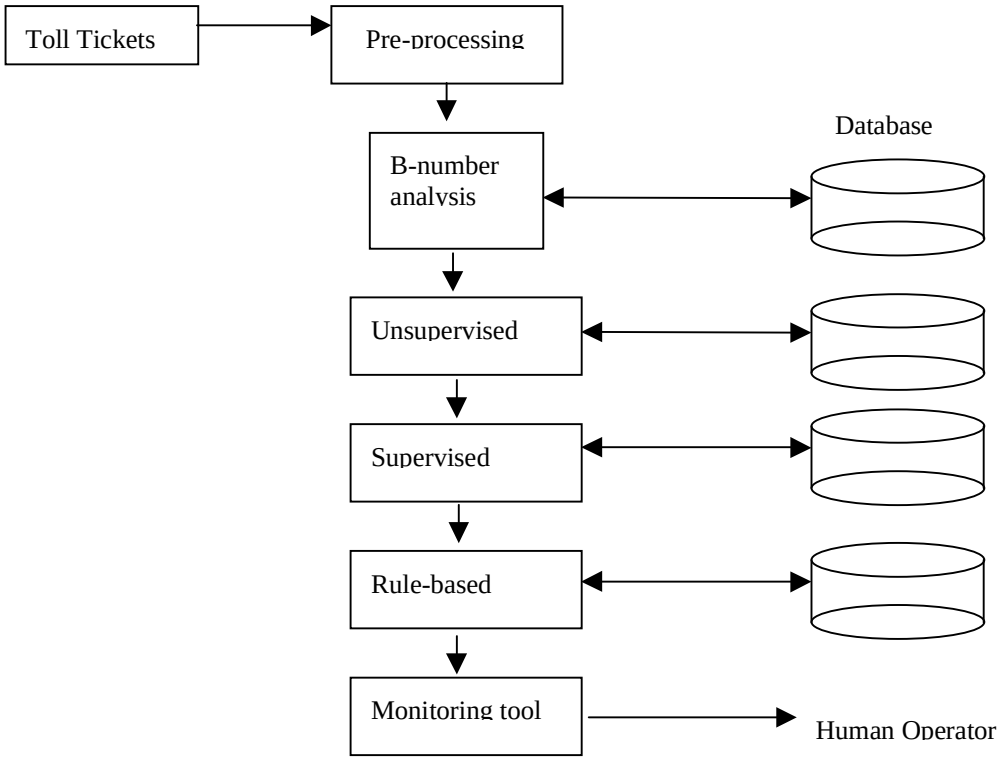

Figure 1.3.1 - The proposed serial integration of the separate fraud detection tools

The demonstrations and trial for the fraud detection tools serve as the basis to meet the main objective of this project, which can be summarized as the study of the feasibility and acceptability of new and advanced security features in existing and future personal communication networks.

The technical feasibility of the developed mechanisms is displayed through the demonstrations. Following this, the acceptability of the security features by users and network operators will be measured during the trial. The trial also provides the means to measure the integrated tools' performance in a simulated real-time network

environment. The outcome of the analysis will give an indication of their performance and of the percentage of frauds that can be identified. Finally, the user interface will be in the form of screen displays and automated reporting facilities provided by the artificial intelligence systems.

Section **2.3** Description of Fraud Detection Trials is structured as follows:

- In **2.3.1** a technical description of the separate components - as they will be put together in the integrated system - is given.
- Paragraph **2.3.2** then describes the datasets that will be used in the trials and to what criteria the integrated system will be tested. The performance of the integrated tool will be evaluated in two senses. On the one hand, the classification performance of the combined tool will be measured through the Receiver-Operator-Characteristic Curve. On the other hand, the usability of the tools' alarms will be investigated from the perspective of the Network.
- Paragraph **2.3.3** presents a high level description of the fraud detection trial configuration. The (software) architecture is discussed and a thorough description of the graphical user interface for the monitoring tool is given.
- Paragraph **2.3.4** provides the trial plan.

2. Trial and Demonstration Specifics

2.1 Authentication Trial

2.1.1 Technical Description

2.1.1.1 Authentication framework

One of the objectives of the authentication framework is "to provide a flexible procedure for user-network authentication" [D05]. This procedure allows a number of different mechanisms and algorithms to be incorporated, with the ability to migrate smoothly from one mechanism to another.

The flexibility is achieved by letting the user (represented by a smart card based UIM), the Network Operator and the Service Provider negotiate the authentication mechanism to be used. A list of acceptable mechanisms will need to be maintained so that different entities can identify and implement the mechanisms they require.

The defined procedures allowing negotiation only involve the user, the Network Operator and the Service Provider. The user is represented by the smart card based UIM. The terminal, which is separate from the user, is not involved in the negotiation. This means that the terminal must be protocol independent. In other words, the authentication protocol and the algorithms necessary to support the authentication must be implemented on the smart card.

Implementing the complete security functionality (protocols and algorithms) on the smart card is not considered feasible at the moment. Therefore the idea of *"Protocol Descriptors and Interpreters in Mobile Terminals"* has been raised in ASPeCT [D11]. The terminal would contain a command interpreter and a list of commands, or a program would be downloaded from the card when it is inserted into the terminal. In this way, the terminal would "learn" the authentication protocol.

This principle is not further elaborated in ASPeCT because it is outside the scope of the project. In the ASPeCT trial, the knowledge of the protocols is "hard-coded" in the terminals, while the algorithms are implemented in the smart card.

Negotiation of the authentication mechanism will be done when EXODUS initiates the authentication framework for the first time between the network and the user. This is not necessarily during user registration since it may be possible that EXODUS does not initiate authentication during user registration. In this case, no negotiation of the authentication mechanism will have occurred. Instead, EXODUS may initiate authentication during call set-up, thereby triggering the negotiation phase.

Note that when a user roams from one network to another and back to the first, and the authentication framework is initiated in each network, then the negotiation phase will also be started the second time that the user enters the first network. In other words, the user will store only one entry with the Network Operator Identification and the authentication mechanism to be used in that network.

It should be noted that different authentication protocols may be used for different users registered on the same terminal.

In order to achieve the negotiation mechanism, a number of procedures are defined in [D05].

2.1.1.1.1 Procedure P1: User - Network Operator authentication capability agreement

User and Network Operator inform each other of their respective authentication capabilities, and subsequently agree the mechanism to be used during their interaction. Exchanges will not include the user's identity, instead an *authentication capability class* indicator would be sent. Note that the user's identity is not sent at this stage, since identification of a user's authentication capability class doesn't require that the visited Network Operator knows the identity of the user. The authentication capability class would be used to categorise users according to their capabilities regarding user-network authentication. A particular class would identify a large

collection of users having the same authentication capabilities: they might have the same version of UIM, for example.

On receiving the user's authentication capability class and Service Provider Identity, the Network Operator checks to see if it has previously interrogated the Service Provider on the authentication mechanisms supported by that authentication capability class. If not, then the Network Operator must run P2 with the user's Service Provider, as described below.

Once the Network Operator and the Service Provider have agreed a mechanism for use with users in a particular authentication capability class, the Network Operator will instruct the user to perform the agreed mechanism.

2.1.1.1.2 Procedure P2: Network Operator - Service Provider authentication capability agreement

Service Provider and Network Operator interact in order to negotiate the user-network authentication mechanism to be used, based on the capabilities and preferences of the entities involved. Specifically, the Network Operator requests information from the user's Service Provider to identify the authentication capabilities possessed by the user, according to his authentication capability class. The Service Provider will respond with the user's authentication capabilities and may also send instructions to the Network Operator to request the use of a preferred mechanism. On receiving this information the Network Operator will make a decision based on the capabilities and preferences, before sending the identity of the prescribed mechanism to the user.

2.1.1.1.3 Procedure P5: User - Network authentication

In ASPeCT Deliverable D2 [D02], three authentication mechanisms are defined.

They all have variants for *"new registrations"* and *"current registrations"* [1]. The new registration variant is used when a user enters a Network Operator's domain for the first time. The current registration variant is used when the user is already known to the Network Operator with whom the user is authenticating.

Registration must be understood here as registration with the Network Operator's authentication centre.

Delegation of user-network authentication to the Network Operator is likely to be required in UMTS in order to reduce the signalling requirements between Network Operator and Service Provider, such that authentication data need only be sent to the Network Operator as part of each new registration[2]. However, since it is the Service Provider that stores the user-related information, the Service Provider must have some involvement in the mechanism, if only to pass or register authentication data with the Network Operator in order to let the Network Operator perform authentication on the Service Provider's behalf, or to delegate (off line) the authority to perform authentication to an approved Certification Authority.

Therefore, depending upon the particular authentication mechanism to be employed for new registrations, either the Service Provider will have a direct interaction with the visited Network Operator, or a Certification Authority approved by the Service Provider will act as an authentication proxy on behalf of the Service Provider.

In the ASPeCT (demonstration and) trial, the Siemens public key mechanism (protocol A and B) and the Royal Holloway secret key mechanism will be implemented.

---

[1] The Siemens proposal for authentication defines Protocols A, B and C. Protocol A is the case where user and Network Operator share each other's public key. In protocol B and C, certificates on each other's public keys are exchanged. In the ASPeCT trial and demonstration protocol A is used for current registrations, protocol B is used for new registrations.

[2] ASPeCT implements two authentication mechanisms. In the Siemens mechanism, protocol B, there is no signalling between Network Operator and Service Provider. In the Royal Holloway mechanism, the Service Provider is interrogated for new registrations.

2.1.1.1.4 Message flow for procedures P1 and P2

The Network Operator sends an initial message to the user instructing him to start the authentication framework. This initial message will contain the Network Operator Identity. The User will notice that it hasn't agreed on an authentication mechanism to be used with the Network Operator.

The user then sends an initial authentication request message to the Network Operator - this will include the user's Service Provider, authentication capability class, but not his identity (nor temporary identity). The Network Operator checks if it knows the user's authentication capability. If it doesn't, the Network Operator sends the user's authentication capability class to his Service Provider. The Service Provider will respond by providing the Network Operator with the specification of that particular authentication capability class - this will include a list of authentication mechanisms the user is capable of handling. The Network Operator will then choose one of the authentication mechanisms, based on the received list of authentication mechanisms and the mechanisms supported by the Network Operator. The Network Operator sends the identity of the prescribed mechanism to the user. The real authentication is then started by the user in procedure P5.

Figure 2.1.1 illustrates the case where the Network Operator doesn't know the received authentication capability class.

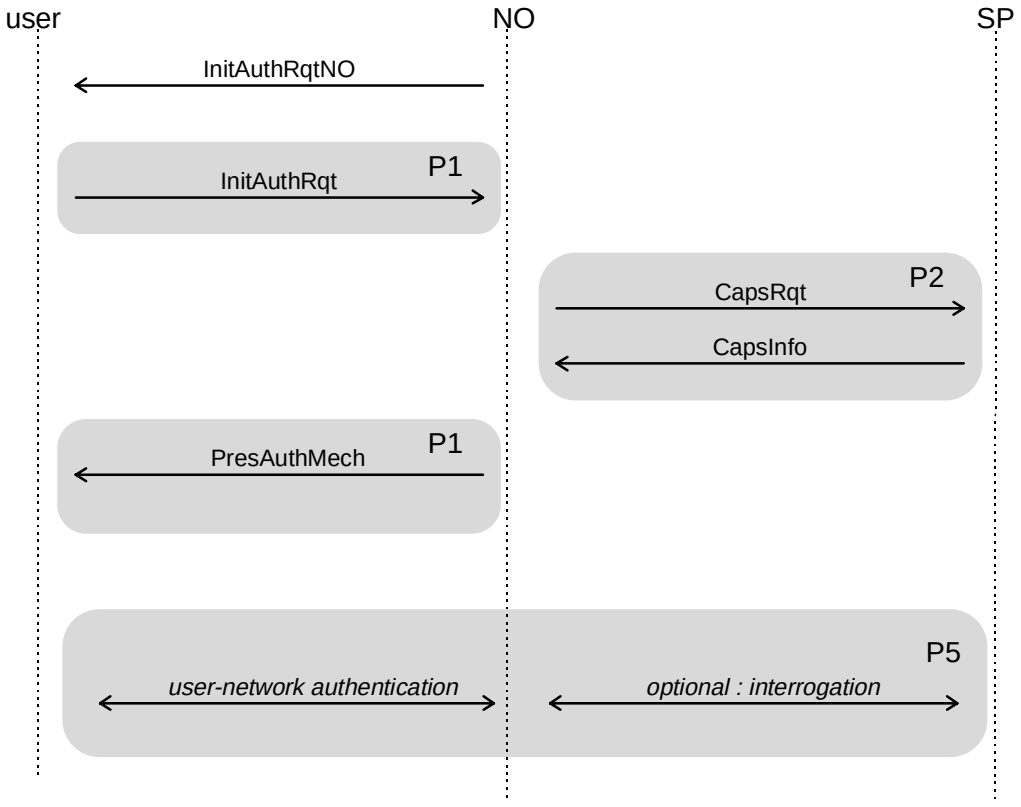

Figure 2.1.1 - Message flow diagram for procedures P1 and P2: negotiation

2.1.1.2 Authentication protocols

In the context of ASPeCT, authentication means mutual authentication between the user and the network (Network Operator and/or Service Provider).

A distinction can be made between two variants of each protocol: authentication for new registrations and authentication for current registrations.

- Authentication for a new registration will always follow immediately on a negotiation procedure.
- Authentication for a current registration will be started when EXODUS initiates the ASPeCT authentication framework and authentication for new registrations has already been performed within the same network.

ASPeCT implements the Siemens public key mechanism (protocol A for current registrations, protocol B for new registrations) and the Royal Holloway Secret key based mechanism (current and new registrations). They both consist of three messages between the user and the Network Operator. The Royal Holloway mechanism exchanges two additional messages between the Network Operator and the Service Provider for new registrations.

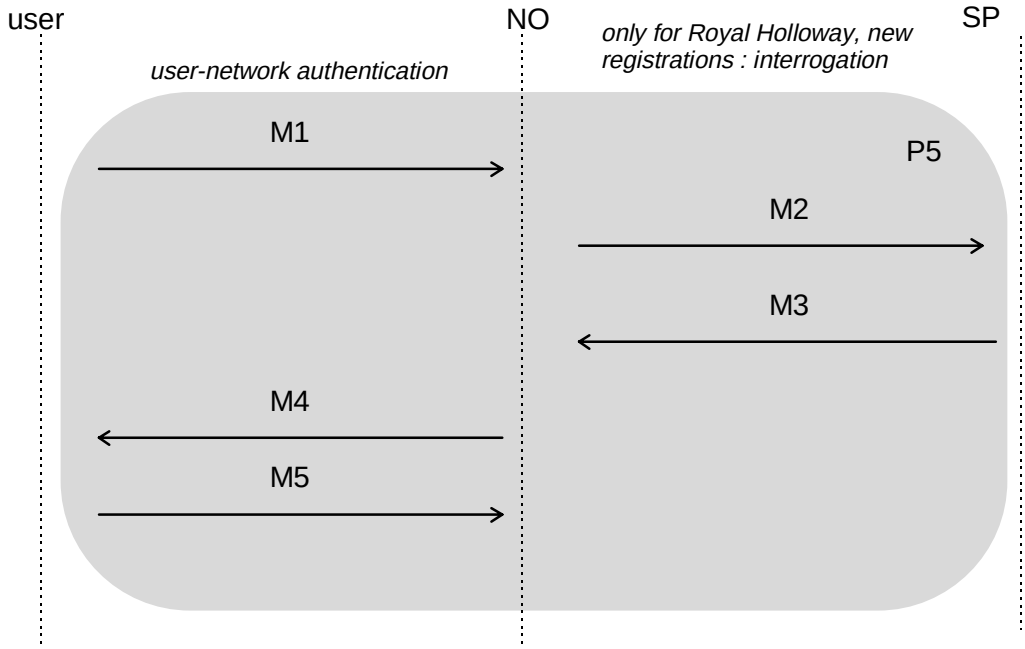

Figure 2.1.2 - Message flow for procedure P5: authentication

2.1.1.2.1 Siemens Authentication protocol

The Siemens protocol is a public key based authentication mechanism defined by Siemens AG and submitted for standardisation. There exist three variants of the protocol. In the following sections only one variant is described, which allows authentication of a user to a network, without the need that they share certificates of each other. This variant is applied when new registration of a user in a network occurs.

The **goals** of the protocol are the following:

- mutual explicit entity authentication of User and Network Operator;
- agreement between the user and the Network Operator on a shared secret key K_S with mutual implicit key authentication;
- mutual key confirmation of the User and the Network Operator;
- mutual assurance of key freshness;
- non-repudiation by the User of data sent by the User to the Network Operator and vice versa;
- confidentiality of the identity IMUI of the User on the air interface;

- exchange of certified public keys between User and Network Operator.

The **data** used within the protocol (|| = concatenate):

AUTH$_N$: This value is calculated to authenticate the Network Operator (NO) to the user.

CertN : This is a valid certificate, issued by a certification authority CA, on a public key of the asymmetric signature system of the NO. It is available at the NO.

CertU : This is a valid certificate, issued by a certification authority CA, on a public key of the asymmetric signature system of the user. It is available at the user.

data1, data2 : Those are optional data fields, to illustrate the non-repudiation feature. data2 is not used for the trial; data1 is the TMUI.

id$_{ca}$: This is the identity of the Certification Authority.

id$_n$: This is the identity of the NO.

K$_S$: This is the session key.

g : Generator g, known by the user, NO and SP. g is a generator of a finite group G with module p (p is a prime) in which the Discrete Logarithm Problem is hard.

s : This is the secret key agreement key for the NO. It is linked with g^s (the public key agreement key).

g^s : This is the public key agreement key of the NO.

IMUI : This is the International Mobile User Identity, uniquely identifying the mobile user.

PK_U : This is the public key of the user used to verify signatures from the user.

RND$_U$: This is a random number generated by the user.

RND$_N$: This is a random number generated by the NO.

The **algorithms** used within the protocol:

h1 : This is a one way function. It is used to calculate the session key:
K_S = **h1(**$(g^{RNDU})^s$ || RND_N**)**

h2 : This is a hash function and used to calculate $AUTH_N$.
$AUTH_N$ = **h2 (**K_S**)**

h3 : This is a hash function and used to calculate a hash value before signature calculation.

Sig$_u$: This is a secret **signature** transformation owned by the user.

Ver$_u$: This is a verification algorithm corresponding to the signature algorithm. This algorithm needs the public key (PK_U in this case) as input.

Enc : This is a symmetric encryption algorithm. **Enc(**K, data**)** means that data is encrypted with key K.

Dec : This is a symmetric decryption algorithm. It corresponds to **Enc().**

The following list is **required** (by user and/or NO):

- the user needs the id_{no} ;
- both the user and NO possess the generator g ;
- the NO has secret and public key agreement keys s and g_s ;

- the NO has a valid certificate CertN ;
- the user has a signature transformation Sigu ;
- the user has a valid certificate CertU.

Selection of algorithms:

h1, h2, h3 : RIPEMD-128 [BP95]

$\boldsymbol{Sig_u, Ver_u}$: AMV signature, based on elliptic curves [ISO96a]

Enc, Dec : DES-CBC [ANSI81], [ANSI83]

The following message flow corresponds to this authentication protocol:

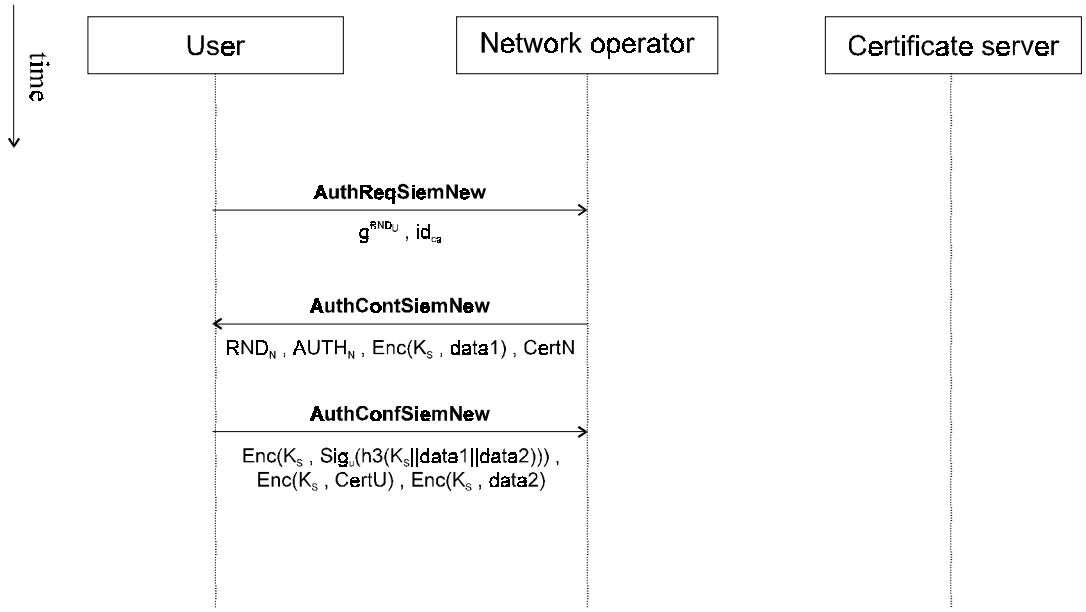

Figure 2.1.3 - Public key authentication mechanism: new registration

The certificate server is not interrogated on-line in this version of the protocol.

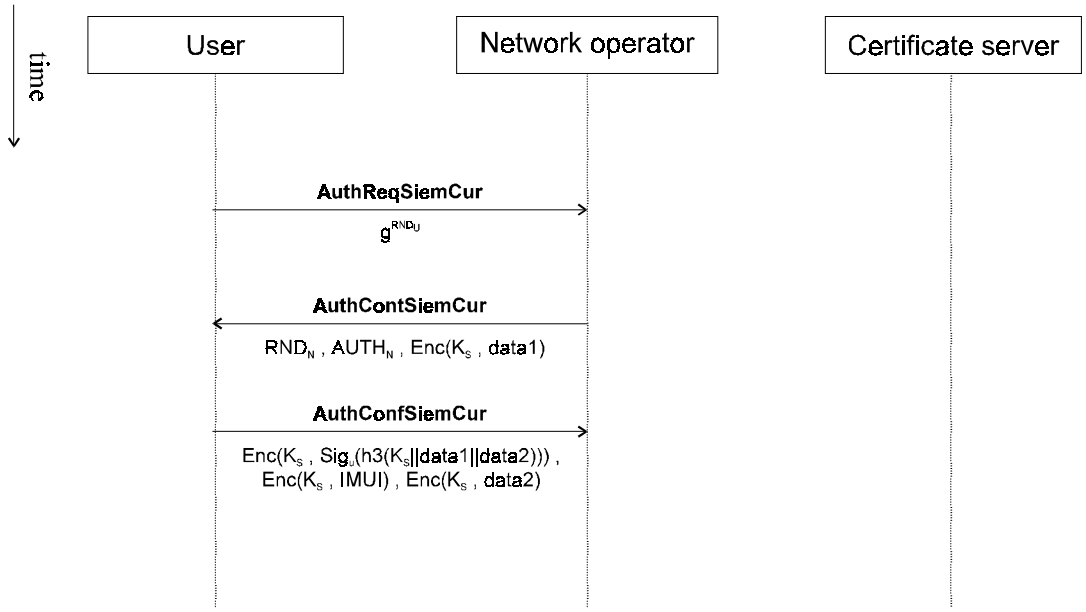

Figure 2.1.4 - Public key authentication mechanism: current registration

2.1.1.2.2 RHUL Authentication protocol

RHUL stands for Royal Holloway University of London. This protocol is an authentication mechanism that is based on a challenge-response mechanism using symmetric keys. In the following sections only one variant is described in detail, to be used when a user and a Network Operator authenticate for the first time.

The **goals** of the protocol are the following:

- mutual explicit entity authentication of User and Network Operator;
- authentication from the Service Provider to the user;
- agreement between the user and the Network Operator on a shared secret key K_S with mutual implicit key authentication ;
- mutual key confirmation of the User and the Network Operator;
- mutual assurance of key freshness;
- establishment of a new user-Network Operator secret key;
- confidentiality of the identity IMUI of the User on the air interface and to the Network Operator, by means of temporary identities.

The **data** used within the protocol (|| = concatenate):

$AUTH_N$: This value is calculated to authenticate the Network Operator (NO) to the user.

$AUTH_S$: This value is calculated to authenticate the Service Provider (SP) to the user.

$AUTH_U$: This value is calculated to authenticate the user to the Network Operator.

$CIPH_N$: This is data used to encrypt the $TMUI_N$'. So even the temporary user identity is enciphered over the radio interface.
Note: It is the result of the C_U algorithm.

$CIPH_S$: This is data used to encrypt the $TMUI_S$'.

K_{NU} : This is a secret key used for authentication that the user shares with the Network Operator.

KO : This is the key offset (used to calculate $CIPH_S$).

K_S : This is a secret key used for encryption and decryption of all the data over the air interface after the authentication protocol ended. The user shares the key with the Network Operator.

K_{SU} : This is a secret key used for authentication that the user shares with the Service Provider.

$NOID$: Network Operator identity.

RND_U : This is the random number generated by the user.

RND_N : This is the random number generated by the Network Operator.

$TMUI_S$: The Service Provider assigns a temporary user identity, used for identification of the user towards the Service Provider. When an accent is added (e.g. $TMUI_S$'), it means that the identifier (here the $TMUI_S$) is a new one.

$TMUI_N$: The Network Operator assigns a temporary user identity, used for identification of the user towards the Network Operator.

The **algorithms** which are important for the **user** (UIM/terminal):

A_K : This hash algorithm generates a session key K_S (for enciphering data over the radio interface).
$K_S = \mathbf{A_K(}\ K_{NU}\ ,\ RND_U \parallel RND_N \parallel TMUI_N'\ \mathbf{)}$

A_N : This hash algorithm generates a user-Network Operator key K_{NU}.
$K_{NU} = \mathbf{A_N(}\ K_{SU}, NOID\ ,\ KO\ \mathbf{)}$

A_S : This hash algorithm generates $AUTH_S$.
$AUTH_S = \mathbf{A_S(}\ K_{SU}, RND_U \parallel TMUI_S'\ \mathbf{)}$

A_U : This hash algorithm generates $AUTH_N$ or $AUTH_U$.
$AUTH_N = \mathbf{A_U(}\ K_{NU}\ ,\ RND_N \parallel RND_U \parallel TMUI_N'\ \mathbf{)}$
$AUTH_U = \mathbf{A_U(}\ K_{NU}\ ,\ RND_U \parallel RND_N\ \mathbf{)}$

C_U : This hash algorithm generates a data string $CIPH_N$ or $CIPH_S$.
$CIPH_N = \mathbf{C_U(}\ K_{NU}\ ,\ RND_U\ \mathbf{)}$
$CIPH_S = \mathbf{C_U(}\ K_{SU}\ ,\ RND_U\ ,\ KO\ \mathbf{)}$

The following list is **required**:

- the user has a temporary identifier (assigned by the Service Provider during a previous registration), which is unique to the registered user and only known by the Service Provider and the user;
- the user and the Service Provider share a unique secret key.

Selection of algorithms:

A_K, A_N, A_S, A_U, C_U: RIPEMD-128

The following message flow corresponds to this authentication protocol:

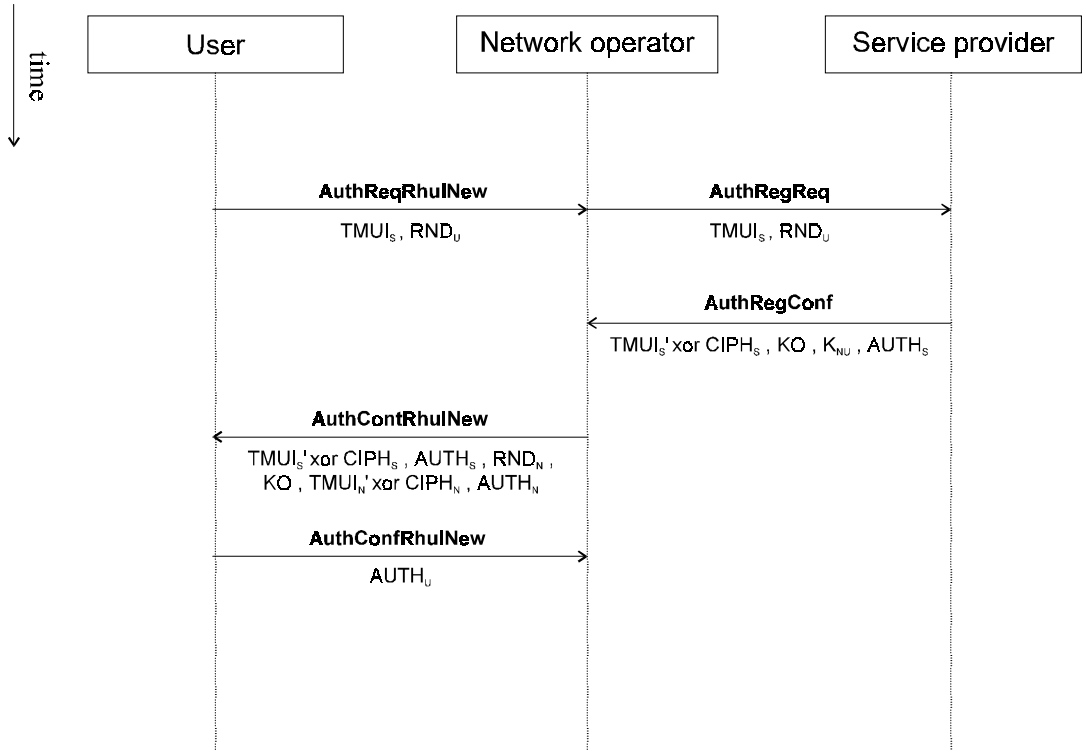

Figure 2.1.5 - Rhul, new registrations

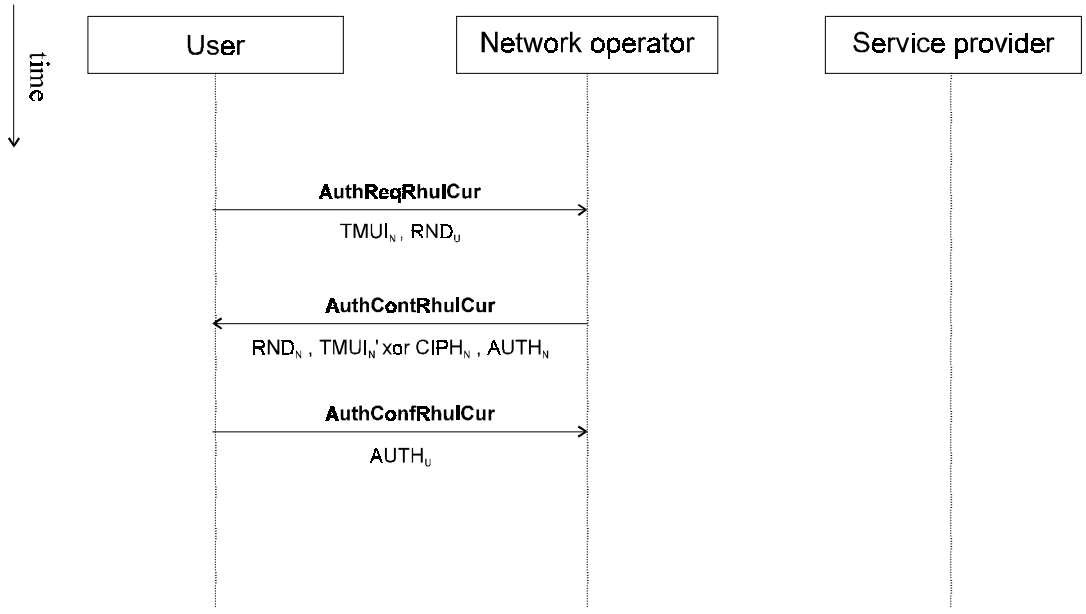

Figure 2.1.6 - Rhul, current registrations

2.1.2 Trial Configuration and Methodology

2.1.2.1 Integration with EXODUS platform

As explained above, ASPeCT has two types of interface with the EXODUS platform (see Figure 1.1.2). The first type connects the Authentication Centre (AuC) to the EXODUS SCP. The other interface type allows the EXODUS terminal (DECT or fixed broadband) to communicate with the smart card based UIM connected to the terminal. These interfaces allow the exchange of ASPeCT mutual authentication messages between the smart card and the AuC, or between two AuCs, over the UMTS network.

The two following sections present the details of the interfaces. The third section details the message flow. The last section shows the integration test scenarios.

2.1.2.1.1 ASPeCT - SCP Interface

The interworking between the AuC PC and the SCP Sun workstation is realised via an Ethernet 802.3 connection, and using the TCP/IP protocol, with (connection-oriented) stream sockets. A client/server model is used for the TCP/IP implementation between the AuC and the SCP, where the AuC acts as the server. The SCP has to connect to the IP address of the AuC, at port 6543.

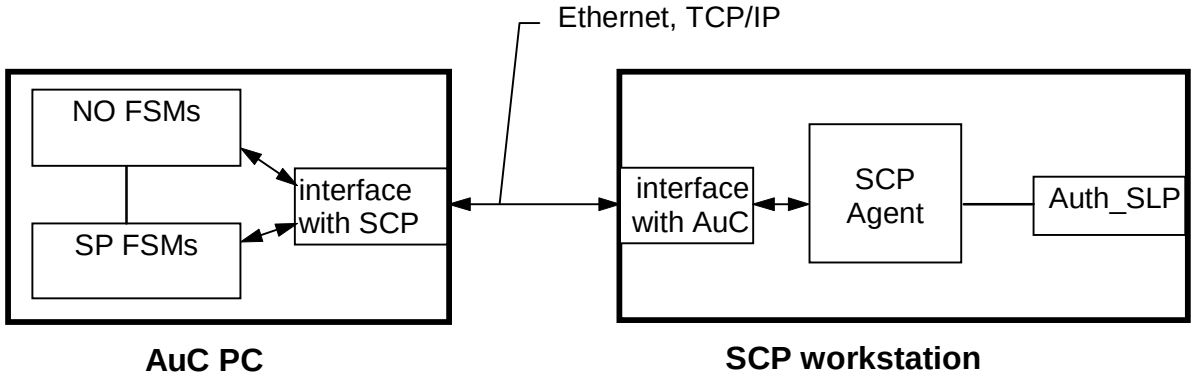

Figure 2.1.7 - Client/server model for the AuC-SCP connection

2.1.2.1.1.1 ASPeCT - SCP Messages

Five types of messages have been defined for communication between the ASPeCT AuC and the EXODUS SCP (see Figure 2.1.8 and Figure 2.1.9). These messages allow the AuC to pass ASPeCT Protocol Data Units (PDUs) over the EXODUS network, which will carry them in two types of EXODUS primitives:

- `Authenticate`: for communication between the AuC and ASPeCT authentication software in the terminal;
- `HandleInformationRequest`: for communication between Visited AuC and Home AuC.

In some cases, ASPeCT PDUs exceed the 58-byte length limit for the DECT air interface. This can occur in both directions. A segmentation scheme has been implemented to take this into account, without interference of the SCP.

Message	TranId	AuthResult	NoId	Destination	SegFlag	SessionKey	AspectPDU
StartAuthReq	*		*				+
AuthResp	*	+					*
SendNextSegment	*						

Figure 2.1.8 - SCP-to-AuC Communication (non optional, + optional)*

Message	TranId	AuthResult	NoId	Destination	SegFlag	SessionKey	AspectPDU
AuthReq	*			*	*		*
StartAuthResp	*	*				+	+

Figure 2.1.9 - AuC-to-SCP Communication (non optional, + optional)*

2.1.2.1.1.2 Description of AuC-SCP Messages Data Elements

TranId

This data element is a unique transaction identification number allocated by the SCP Agent Function at the start of a dialogue with the AuC, and will be used in all messages within that dialogue. This will enable the SCP to carry out more than one dialogue with the AuC at a time.

Type INTEGER

AuthResult

This data element identifies to the SCP the result (success, or different types of failure) of the authentication of the user at the end of the dialogue.

Type ENUMERATED

NoId

The Network Operator identity is supplied by the SCP only in the StartAuthReq operation, at the beginning of the dialogue with the AuC. The AuC inserts it in the ASPeCT PDU and sends it to the terminal, where it is used to assist in deciding whether negotiations are to be done or not, i.e. whether to run P1 or P5.

Type OCTET STRING

Destination

The SCP uses the destination element to decide if the AuthReq message from the AuC is to be routed to the terminal, or to another SCP/AuC. A 0 value will indicate to send to the terminal, while a strict positive value indicates to send to the other SCP/AuC (this is done this way since there are only two SCPs in the EXODUS system).

Type ENUMERATED

SegFlag

Segmentation of an ASPeCT PDU is indicated by this data element. It can have three values:

- *not_segmented* indicates a non-segmented aspect-PDU in the message;
- *segmented_notlast* indicates that the message contains a segment (but not the last one) of an aspect-PDU;
- *segmented_last* indicates that the message contains the last segment of a segmented aspect-PDU.

This element is used by the SCP to generate the correct termination signal to the MB-SSP FSM. If the SCP receives a message from the AuC where the SegFlag is set to *segmented_notlast*, it forwards the ASPeCT PDU to its destination and then asks the AuC to send the next segment (`SendNextSegment` message).

Type ENUMERATED

AspectPDU

These are ASPeCT Authentication messages and are carried as PDUs in EXODUS primitives. They are passed through the SCP to or from their destinations unhindered.

Type OCTET STRING

SessionKey

This identifies the session key KeyS, which is calculated during the authentication protocol.

Type OCTET STRING

2.1.2.1.2 ASPeCT - Terminal Interface

2.1.2.1.2.1 General

The interface between ASPeCT and EXODUS on the terminal PC is a one-way procedural interface. The EXODUS software will call procedures in the ASPeCT software.

2.1.2.1.2.2 Interfaces needed for the Migration Trial

In the following sections a short description is given of the interfaces between ASPeCT and EXODUS needed for the ASPeCT Migration Trial.

The EXODUS software will call procedures to read data stored on the UIM, to read the status of the UIM, to transmit ASPeCT PDUs to the ASPeCT software and to read ASPeCT PDUs that have to be sent to the AuC.

The last section gives an example of a flow for an authentication and indicates the called procedures.

2.1.2.1.2.2.1 Create_User, Destroy_User

Before the other procedures can be used, the Create_User procedure has to be called. It initialises the ASPeCT software and the smart card reader for further processing. At the end, the Destroy_User procedure has to be called to close the smart card reader.

Input None

Output Yes : user successfully created/destroyed

No : problem with creating/destroying user

2.1.2.1.2.2.2 Card_Present

This procedure checks whether a smart card is present in the intelligent card reader. The EXODUS software should call this procedure at regular intervals, e.g. every 3 seconds.

Input None

Output Yes : a smart card is present

No : no smart card is present

2.1.2.1.2.2.3 Verify_PIN

This procedures checks the access conditions for data stored on the smart card, by verifying PIN Code 1. After a successful presentation, the PIN_1 level remains valid until the smart card is powered off (removed). Any data elements stored in the UIM classified as “PIN_1 protected” can be read or modified according to the respective access conditions for the files.

Note: It is important to distinguish between a PIN typed in at the multimedia terminal which is transparently sent to the UIM and a PIN typed in on the card reader’s display after insertion of the smart card. The trial will employ the second solution.

Input None (PIN_1 has to be entered on the card reader)

Output Yes : PIN_1 code is correct

No : PIN_1 code is wrong

2.1.2.1.2.2.4 Read/Write data stored on the smart card

The following procedures are available to read or write data:

- *Read and Write International Mobile User Number* (17 possible numbers):
  Read_IMUN, Write_IMUN
- *Read International Mobile User Identity*:
  Read_IMUI
- *Read and Write Temporary Mobile User Identity*:
  Read_TMUI, Write_TMUI
- *Read and Write Location Area Identifier and Fixed Point Attachment Identifier*:
  Read_LAI_FPAI, Write_LAI_FPAI
- *Read and Write Language Identity*:
  Read_LI, Write_LI
- *Read and Write Calling Party Number*:
  Read_CPN, Write_CPN
- *Read and Write Service Provider Identity*:
  Read_SPI, Write_SPI

Input Identifier for data to be read / write

Output Data and result of read / write operation

2.1.2.1.2.2.5 Exchange of ASPeCT PDUs between ASPeCT and EXODUS

The EXODUS software receives Authenticate req/ind primitives from the network. These primitives contain ASPeCT PDUs which must be routed to the ASPeCT software. EXODUS calls a procedure *Handle_ASPeCT_PDU* from ASPeCT to deliver the PDU to ASPeCT. The type of terminal (DECT or multimedia) has to be indicated, because the DECT interface requires segmentation (message length limit is 58 bytes). After that the EXODUS software must call a procedure *Get_ASPeCT_PDU* to read the PDU that ASPeCT wants to send to the network. This PDU is put in an Authenticate rsp/conf primitive and sent to the network.

On receipt of an Authenticate req/ind primitive, the EXODUS terminal should mark the user related to the Authenticate primitive as “not authenticated”. The user is marked as “authenticated” after receipt of an ASPeCT PDU together with a positive authentication result parameter.

2.1.2.1.2.2.5.1 Handle_ASPeCT_PDU

This is used to pass the ASPeCT PDUs from the EXODUS software in the terminal to the ASPeCT Software in the terminal.

Input ASPeCT PDU

Terminal_type (DECT or multimedia)

Output Result of the procedure call. Normally this result should be successful. The call may be unsuccessful, e.g. if the ASPeCT software is not configured correctly.

2.1.2.1.2.2.5.2 Get_ASPeCT_PDU

The EXODUS software in the terminal calls this procedure to retrieve ASPeCT PDUs from the ASPeCT software in the terminal.

Note: if the Handle_ASPeCT_PDU procedure passes a non-final segment to the ASPeCT software in the terminal, then any call to Get_ASPeCT_PDU will be unsuccessful until the last segment of this message has been passed to the ASPeCT software and the recomposed message has been processed successfully.

If successive calls to Get_ASPeCT_PDU do not return a positive answer within a certain time limit, the EXODUS software may decide not to call the ASPeCT procedure any longer and abort the ongoing procedure. This time limit is to be set by EXODUS. In these cases, it may happen that ASPeCT writes a message in a buffer which will never be read out. The buffer will be overwritten during a next call to Handle_ASPeCT_PDU.

Input Pointers to store ASPeCT PDU, authentication result and key Ks.

Output Result of procedure call :

OK : successful, other parameters are also available

Not OK : not successful, ASPeCT is still busy or is waiting for another segment

Optional output parameters (only available if the result of the procedure call is OK)

ASPeCT PDU : the ASPeCT PDU which must be sent to the network.

Authentication_Result : This parameter is null as long as authentication is not completed successfully. As soon as authentication is completed successfully for the terminal side, the flag is positive. This can only happen with a message M5 in the ASPeCT PDU parameter.

Ks : This parameter is only filled in when the Authentication_Result flag is positive. It is the shared secret key generated during the authentication. It can be used for subsequent ciphering.

2.1.2.1.2.3 Example of Flow

The following figure (Figure 2.1.10) illustrates an example of a flow for User Registration. Timer1 is the polling timer to check the presence of a smart card in the card reader.

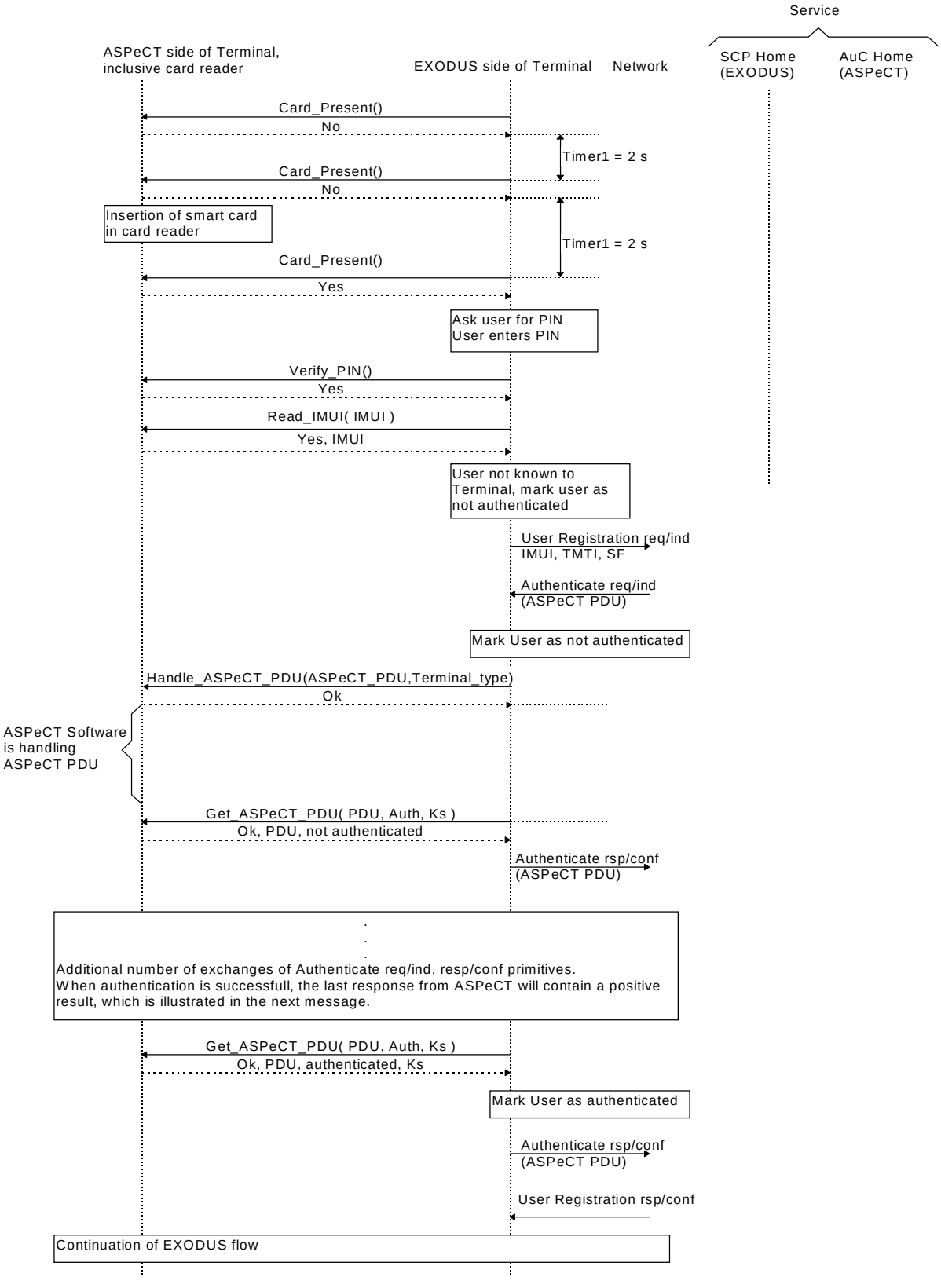

Figure 2.1.10 - Example of flow for User Registration

2.1.2.1.3 Message Flow Description

2.1.2.1.3.1 Routing of ASPeCT messages

ASPeCT messages arriving in the SCP via the EXODUS interface will always be routed to the connected AuC, so there is no routing information needed in the incoming messages.

ASPeCT messages arriving in the SCP via the AuC interface are routed by the SCP to the terminal, or to another SCP/AuC. To distinguish between both destinations, the AuC passes a destination identifier to the SCP, in addition to the ASPeCT PDU.

2.1.2.1.3.2 Start of ASPeCT authentication framework

The SCP associates a unique transaction identifier to each dialogue between the AuC and the user terminal. This identifier may be assigned randomly, but may not be in use by different users at the same time. Every message exchanged between AuC and SCP will be accompanied by the transaction identifier. This identifier is only known by the SCP and the AuC, and *not* by the user at the terminal end.

If EXODUS wants to start the authentication framework, the EXODUS SCP has to send a `StartAuthReq` message to the connected AuC. This message will contain the Network Operator identity as parameter.

The ASPeCT AuC reacts by sending to the SCP an `AuthReq` message, containing the first ASPeCT authentication message `InitAuthRqtNO` in the AspectPDU field, and destined for the terminal. The SCP will put the `InitAuthRqtNO` PDU in an EXODUS `authenticate` req/ind primitive and route it to the terminal.

The ASPeCT PDU `InitAuthRqtNO` will contain the Network Operator identity.

It is then up to the ASPeCT side of the terminal to decide which ASPeCT-procedure to start:

- When the user has not agreed on a mechanism to be used with this Network Operator, then the procedure for negotiation will start (see Figure 2.1.1) followed by the *new registration* procedure for authentication (see Figure 2.1.2).
- When the user has already agreed on a mechanism to be used with this Network Operator, then the *current registration* procedure for authentication will start immediately. If for any reason the authentication was not successful, the procedure can be repeated with the negotiation phase and the *new registration* authentication procedure.

This is illustrated in Figure 2.1.11 and Figure 2.1.12. Both figures illustrate a message flow from the start until the sending of the first ASPeCT message from terminal to network.

In Figure 2.1.11, the ASPeCT authentication framework is triggered during the EXODUS procedure User Registration. The user has not agreed on a mechanism to be used with this Network Operator, and thus initiates negotiation.

After receiving the `UserRegistration` req/ind primitive, the Network Operator sends a `StartAuthReq` message to the connected AuC, with a parameter identifying the Network Operator. The ASPeCT AuC will send a `InitAuthRqtNO` PDU containing the Network Operator identity, encapsulated in an `AuthReq` message, to the SCP, with destination identifier the terminal. The SCP will put the ASPeCT PDU in an `authenticate` req/ind primitive and route it to the terminal.

ASPeCT will then start the negotiation procedure P1.

The example in Figure 2.1.12 is where a Location Update is done. This example assumes that negotiation has already occurred.

After receiving the `LocationUpdate` req/ind primitive, the Network Operator sends a `StartAuthReq` message to the connected AuC, with a parameter identifying the Network Operator. The ASPeCT AuC will send the `InitAuthRqtNO` PDU containing the Network Operator identity, encapsulated in an `AuthReq` message, to the SCP, with destination identifier the terminal. The SCP will put the ASPeCT PDU in an `authenticate` req/ind primitive and route it to the terminal.

ASPeCT will then start the authentication procedure P5.

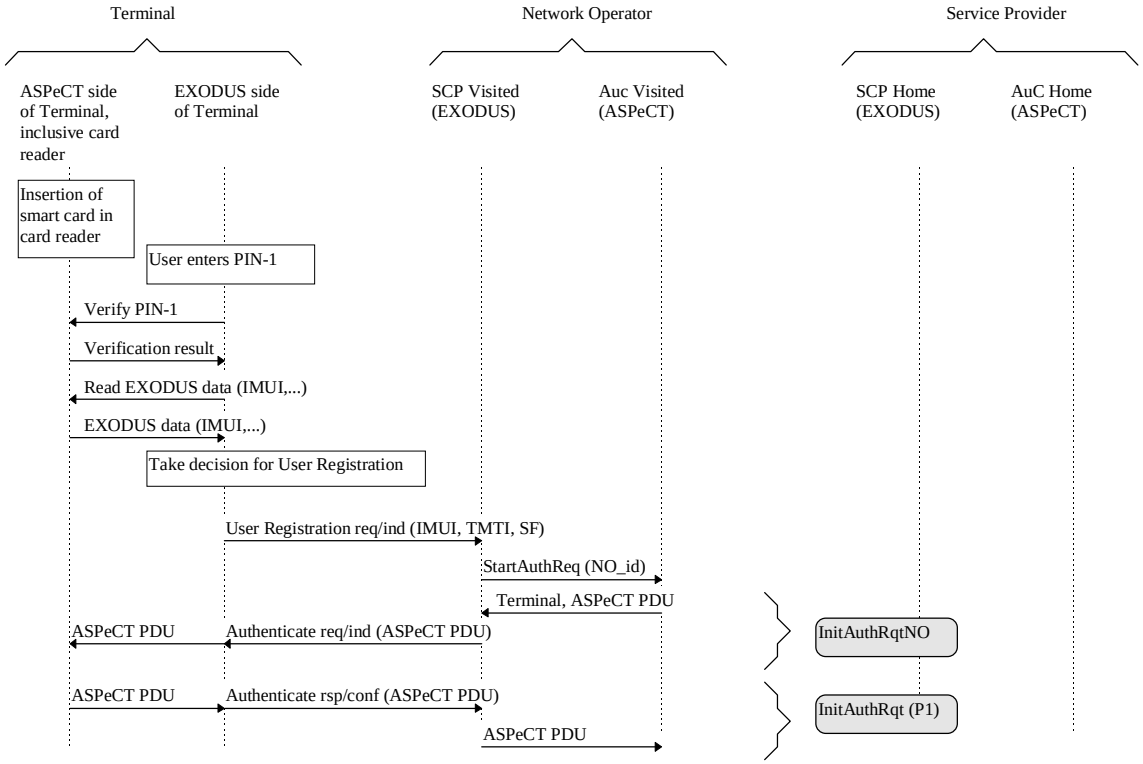

Figure 2.1.11 - Triggering of ASPeCT authentication framework, With Negotiation

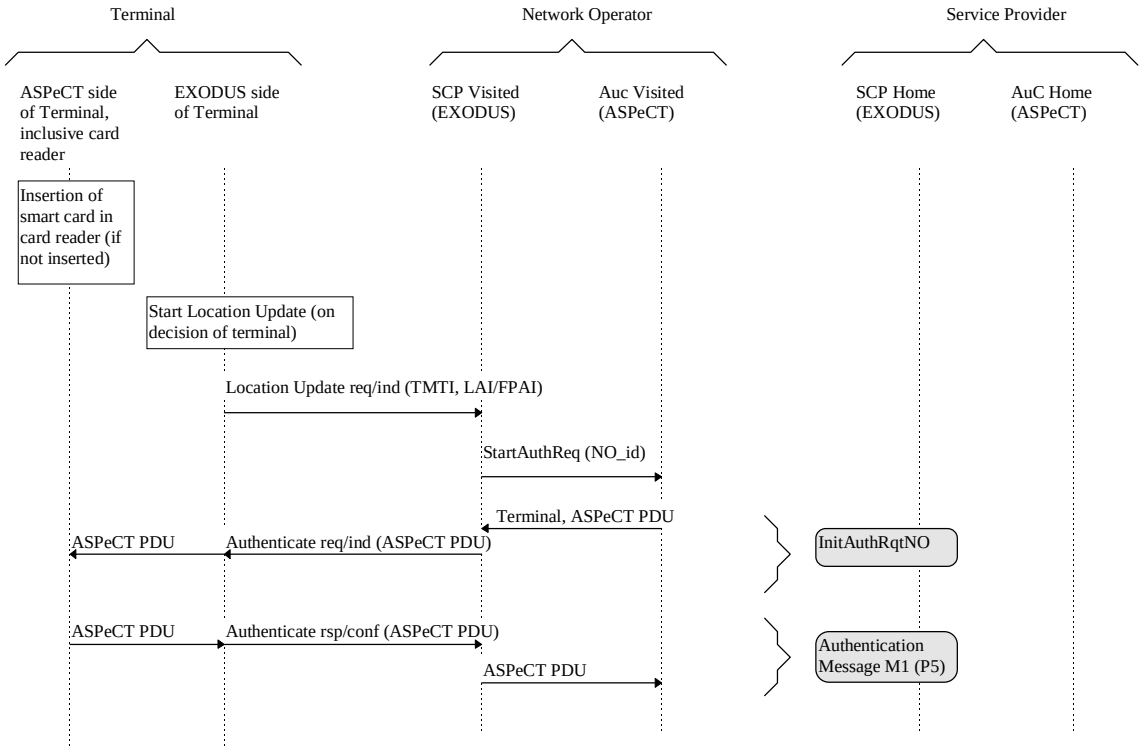

Figure 2.1.12 - Triggering of ASPeCT authentication framework, Without Negotiation

2.1.2.1.3.3 ASPeCT authentication framework

The start of the ASPeCT authentication framework and the interaction with EXODUS has been explained in the previous section. The `InitAuthRqt` PDU will be contained in the `Authenticate` req/ind primitive. The response to this primitive will contain the first ASPeCT authentication framework primitive.

The authentication framework will then be completed by exchanging additional `authenticate` req/ind, resp/conf primitive pairs between the Visited SCP and the terminal, and `HandleInformationRequest` req/ind, resp/conf primitive pairs between the Visited SCP and the Home SCP. These primitives will contain ASPeCT PDUs.

Two examples are illustrated in the following figures:

Figure 2.1.13: ASPeCT authentication framework with negotiation.

Figure 2.1.14: ASPeCT authentication framework without negotiation.

All the primitives contain the actual PDUs needed for ASPeCT. These contents are transparent for the SCP.

Sometimes a destination identifier is included (AuCHome, AuCVisited or terminal). These are cases where the SCP needs to look at the destination identifier and route the primitive.

Note that the interrogation between Network Operator and Service Provider is optional. It depends on the used mechanism. In ASPeCT there will only be interrogation for new registrations using the Royal Holloway mechanism.

If the AuC wants to interrogate the home Service Provider, the AuC will send the ASPeCT message encapsulated in an `AuthReq` message to the SCP (destination is then AuCHome). The SCP routes the message to the home SCP, and this one delivers the ASPeCT message, encapsulated in a `StartAuthReq`

message, to the attached AuC. The Service Provider associated with this AuC then computes the response and returns this as an ASPeCT PDU in a `StartAuthResp` to the SCP, who will pass it to the visited SCP. This one finally returns an answer to the visited AuC, encapsulated in an `AuthResp` message.

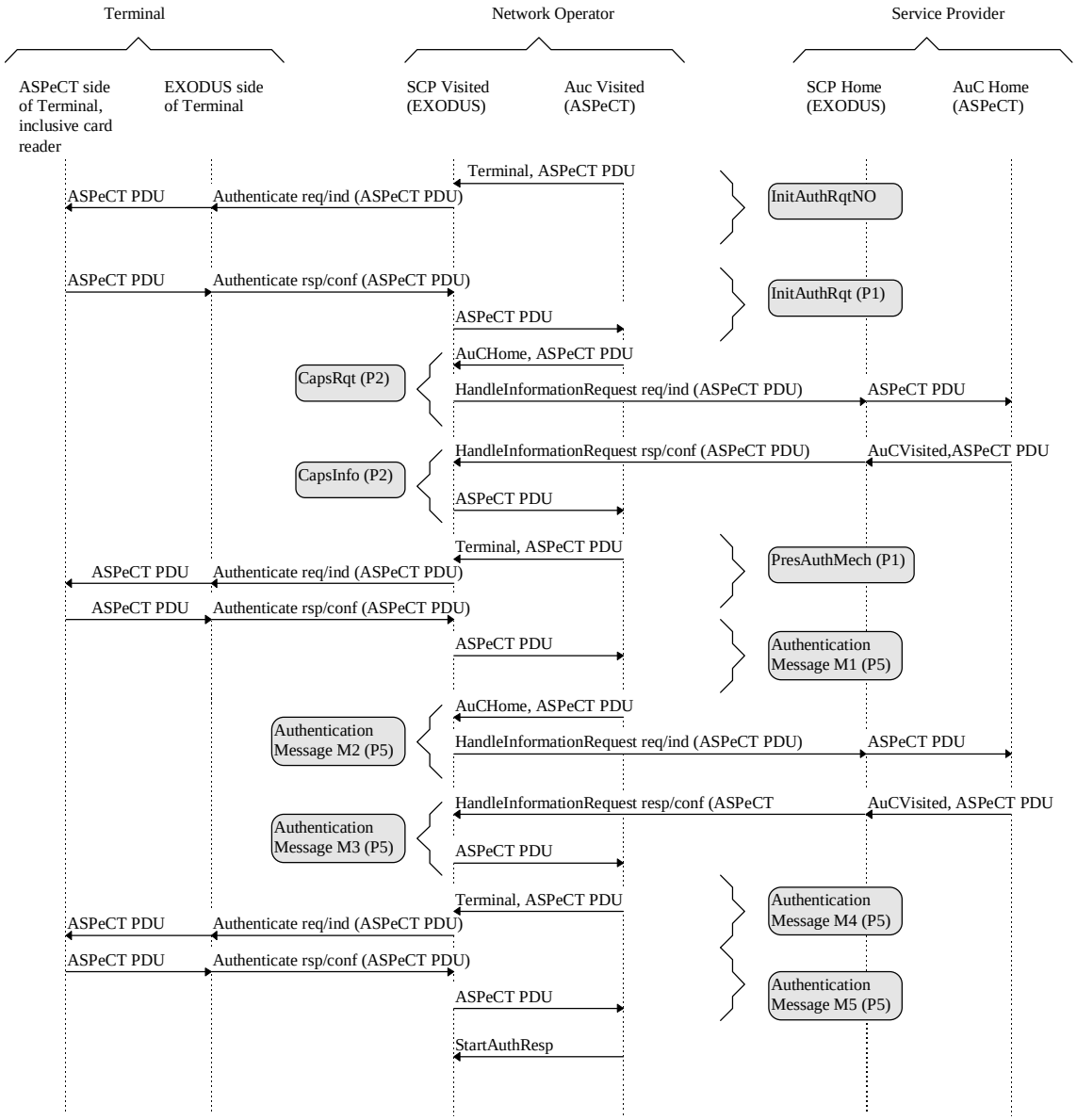

Figure 2.1.13 - Example of ASPeCT authentication framework with negotiation

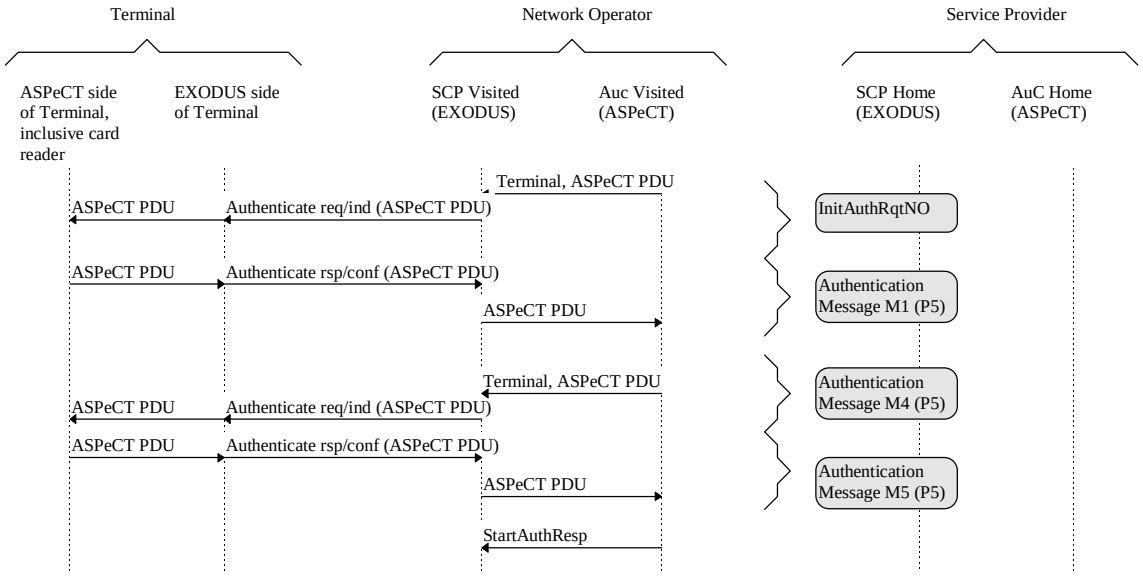

Figure 2.1.14 - Example of ASPeCT Authentication framework, without negotiation

2.1.2.1.3.4 End of ASPeCT authentication framework

Both the EXODUS side of the terminal and the Visited SCP must know the result of the ASPeCT authentication framework and, if authentication was successful, the generated cipher key Ks. The cipher key is generated both by the AuC and the terminal.

For every incoming message from the EXODUS side of the terminal, the ASPeCT side of the terminal will return an ASPeCT PDU and an authentication result; if the authentication was successful, the cipher key will be returned as well. The EXODUS terminal will read the result and key and will only send the ASPeCT PDU to the network.

To inform the Visited SCP about the result of the authentication at the end of the protocol, the AuC sends a `StartAuthResp` message to the Visited AuC. If the AuthResult element of the message indicates success, the SessionKey element will contain the same cipher key, but generated in the AuC.

An example can be found in the following figure.

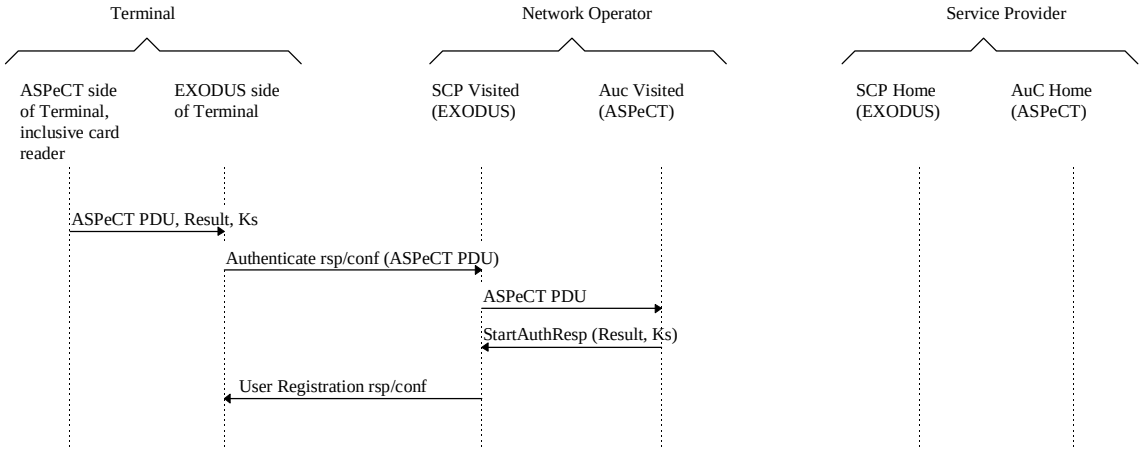

Figure 2.1.15 - End of ASPeCT Authentication Framework

2.1.2.1.3.5 Segmentation of messages

The DECT air interface enforces a 58 bytes length limitation on the exchanged ASPeCT PDU messages, requiring a segmentation mechanism for longer messages. The segmentation occurs on the ASN.1 encoded PDU; the first 58 bytes segment will thus contain the tag, the length and part of the value of the encoded message.

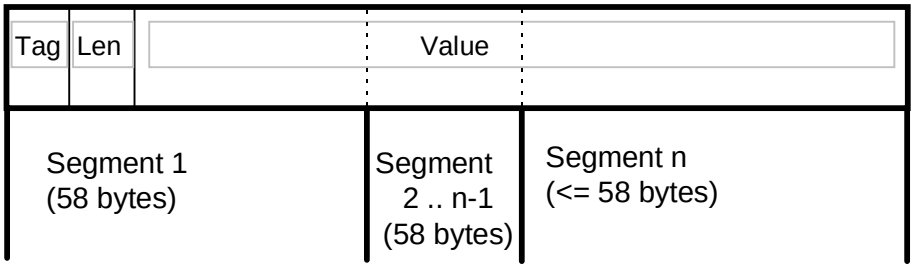

Figure 2.1.16 - ASN. 1 encoded message (tag, length, value)

The following messages are longer than 58 bytes: Authentication Messages M4 and M5 for the Siemens Protocol, new registrations; M5 for Siemens, current registrations; and M4 for Rhul, new registrations.

As the message length limitation does not exist for fixed terminals, the Visited AuC must know in some way the type of terminal which is used. The terminal type will therefore be added as a parameter in the Handle_ASPECT_PDU function, called by the EXODUS side of the terminal, and transmitted to the Visited AuC, in the ASPeCT PDU containing negotiation message P1. This message is only sent during new registrations, but this is the only case where the Visited AuC has to send a segmented message before it receives a segmented message. The AuC can detect an incoming segmented message by comparing the length of the ASPeCT PDU element in the `AuthResp` message from the SCP, with the length field inside the (first segment of the) encoded message.

Segmentation of messages is performed by the ASPeCT applications in a way transparent to EXODUS, by exchanging one or more authenticate req/ind, rsp/conf primitive pairs. The next sections present the information flows illustrating segmentation in both directions.

2.1.2.1.3.5.1 Method for Segmentation in the Service To User Direction

When the ASPeCT software in the AuC sends a segment (not the last one) to the terminal, it waits for the SCP to send a `SendNextSegment` AuC-SCP message, before sending the next segment. The SCP sends the `SendNextSegment` AuC-SCP message to the AuC when it receives an `AuthReq` message with a segmentation flag indicating that the message contains an ASPeCT PDU segment (not the last one). Only when the last segment has been sent, the SCP issues a `Send Component` INAP operation containing the request to monitor for a response.

The segment is delivered to the ASPeCT side of the terminal in the same way complete messages are delivered. In the case of a non-final segment however, the terminal does not return an ASPeCT PDU to the EXODUS side of the terminal.

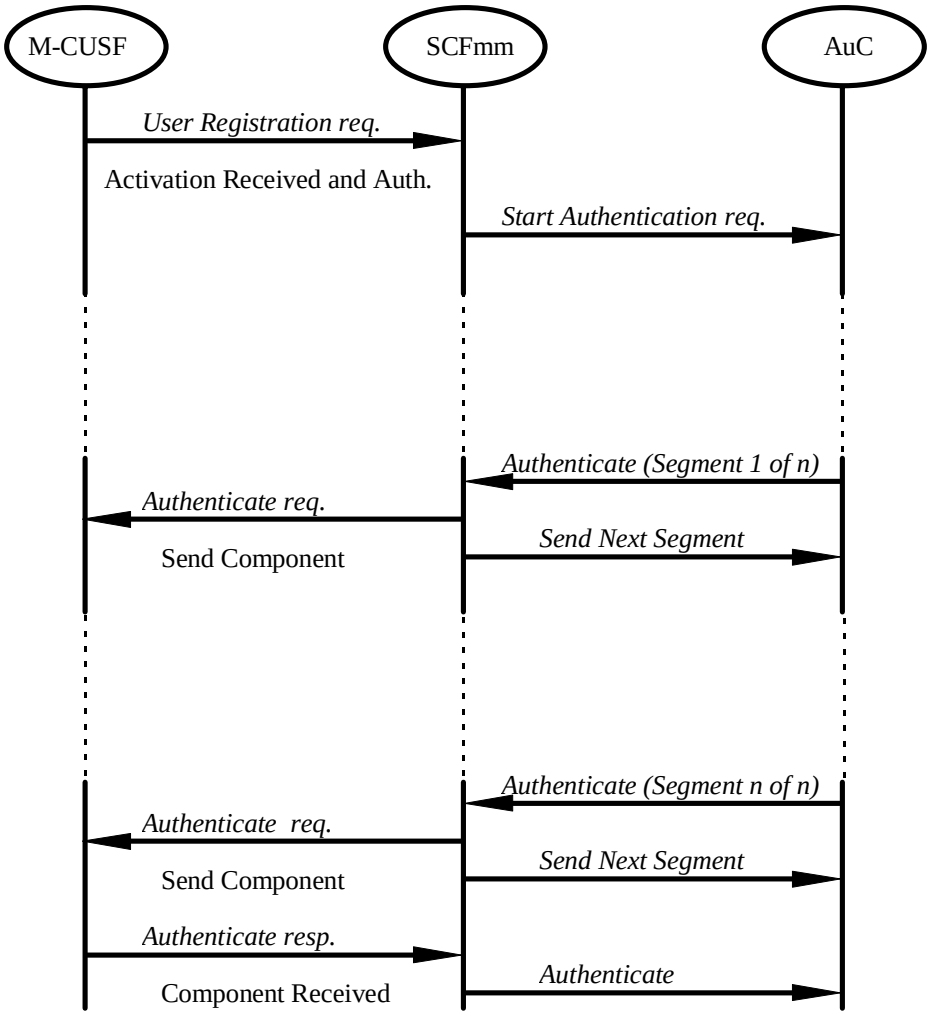

Figure 2.1.17 - Method for Segmentation in the Service to User Direction

2.1.2.1.3.5.2 Method for Segmentation in the User To Service Direction

When the ASPeCT software in the AuC receives a segment (not the last one) from the terminal, it replies with an `ReqNextSegment` ASPeCT PDU. Upon reception of this message, the ASPeCT side of the terminal sends the next segment.

Each `Authenticate` message segment must be individually requested, because each time a segment is received by the M-CUSF, and passed on in a `Component Received` INAP operation, the M-CUSF state machine changes from “Monitoring” to “Waiting for Instructions”. In this state, only operations from the SCF can be received. Therefore, the SCF must specifically request monitoring for receipt of the next segment using a `Send Component` INAP operation.

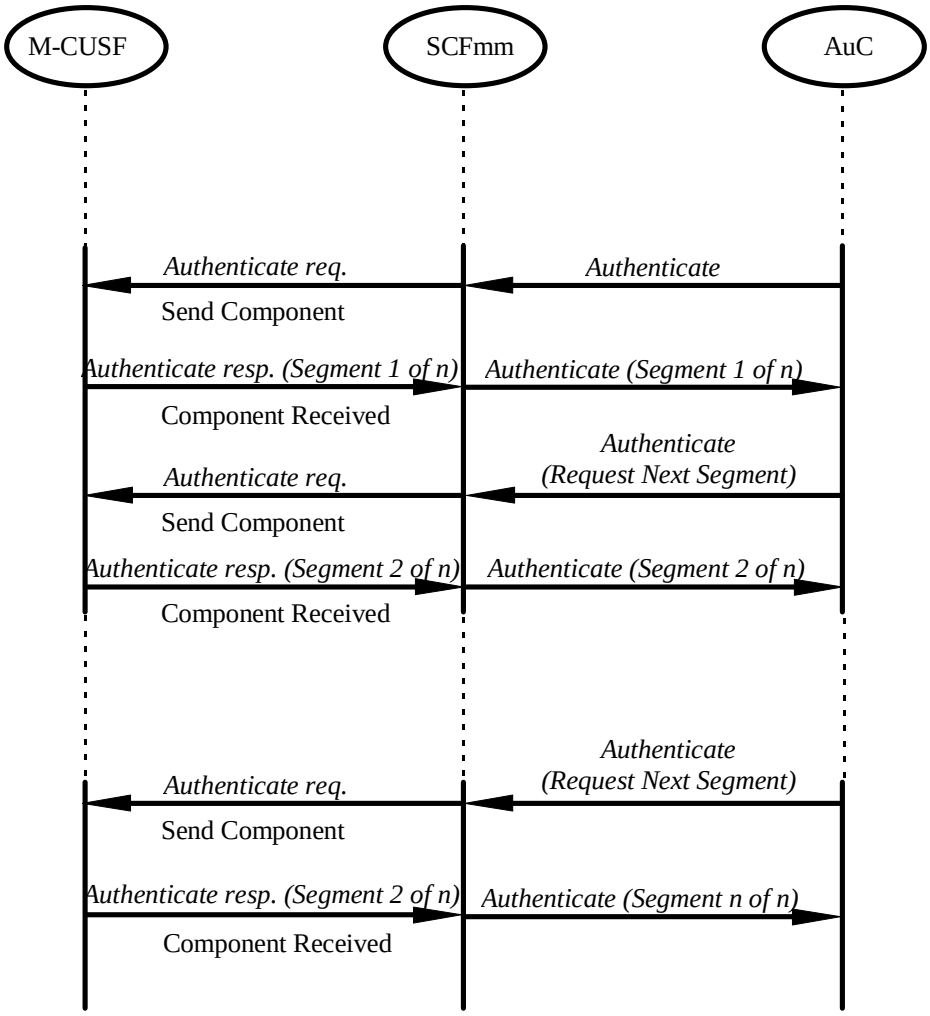

Figure 2.1.18 - Method for Segmentation in the User To Service Direction

The following figure illustrates the segmentation of messages for authentication, taking into account the two different ways of segmentation for user to network and network to user direction.

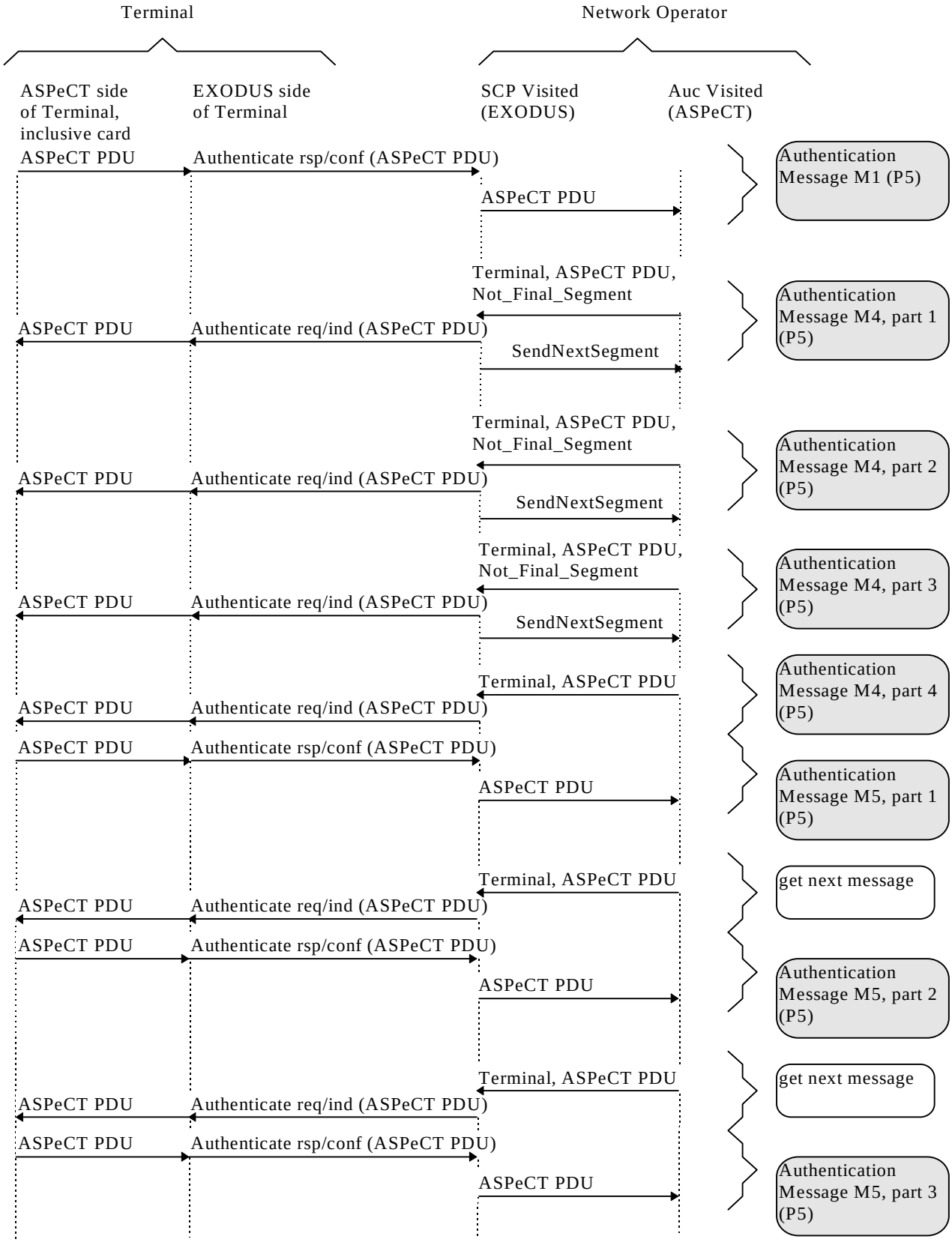

Figure 2.1.19 - Segmentation of long messages

2.1.2.1.3.6 Error Cases

Errors during the protocol can be detected either by the ASPeCT software on the terminal or by the Authentication Centre connected to the visited or the home SCP.

- In case of errors detected by the ASPeCT-side of the terminal, an authentication reject PDU, containing the reason for the rejection, will be sent in the `Authenticate` rsp/conf primitive. The AuC connected to the visited SCP will send a negative `StartAuthResp` primitive to the SCP, containing the reason of rejection in the AuthResult element. The SCP will then send a negative EXODUS primitive to the EXODUS side of the terminal. An example is presented in Figure 2.1.20 where for the case of location update, without negotiation, an error is detected by the terminal in authentication message M4. The flow is preceded by the flow in Figure 2.1.14.

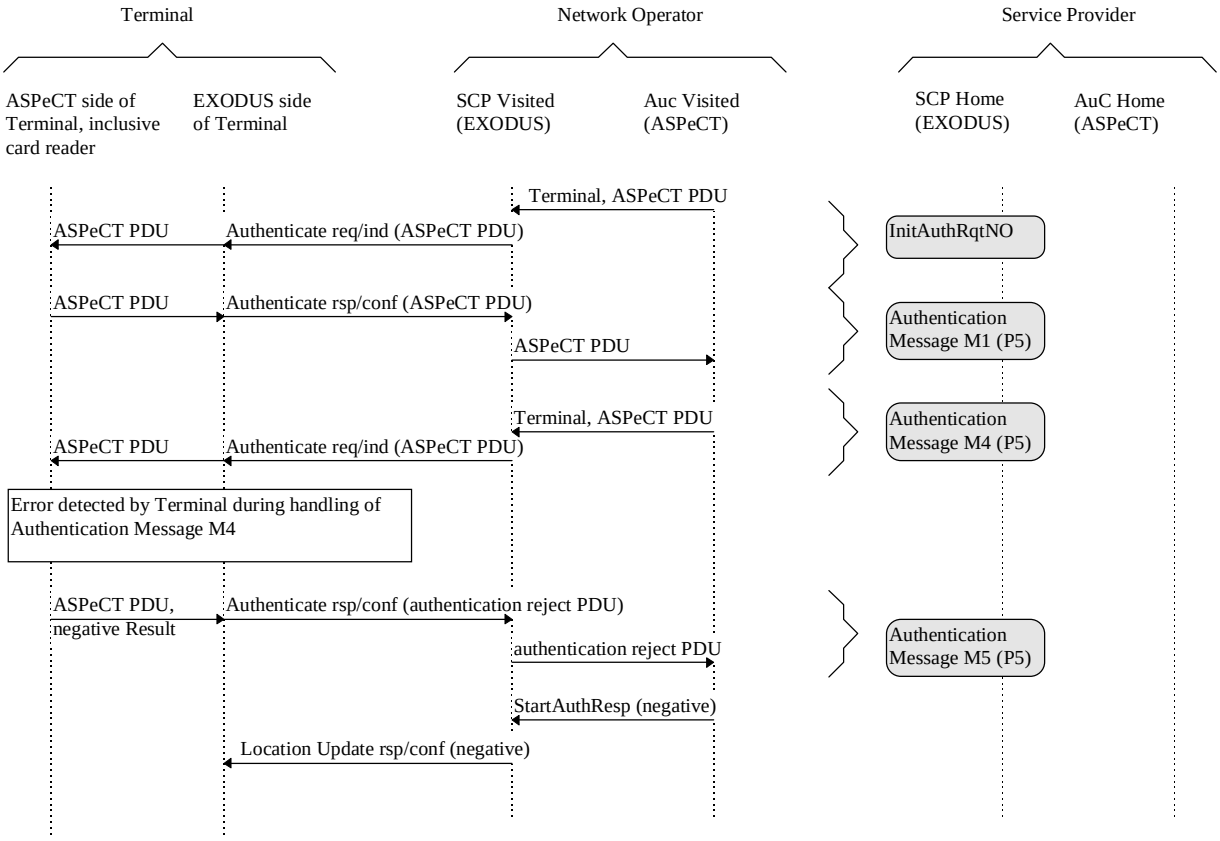

Figure 2.1.20 - Error in protocol detected by Terminal

- In case the errors are detected by the AuC (visited or home) then an additional pair of `Authenticate` req/ind rsp/conf primitives must be sent to the terminal containing the authentication reject PDU. This must be done for two reasons :

  1. The error can be detected by the AuC while the ASPeCT side of the terminal is still in a state waiting for an ASPeCT PDU.
  2. There is no other way to inform the ASPeCT side of the terminal that the authentication failed.
     The example in Figure 2.1.21 is similar to the previous case, but here the error is detected by the visited AuC in authentication message M5.

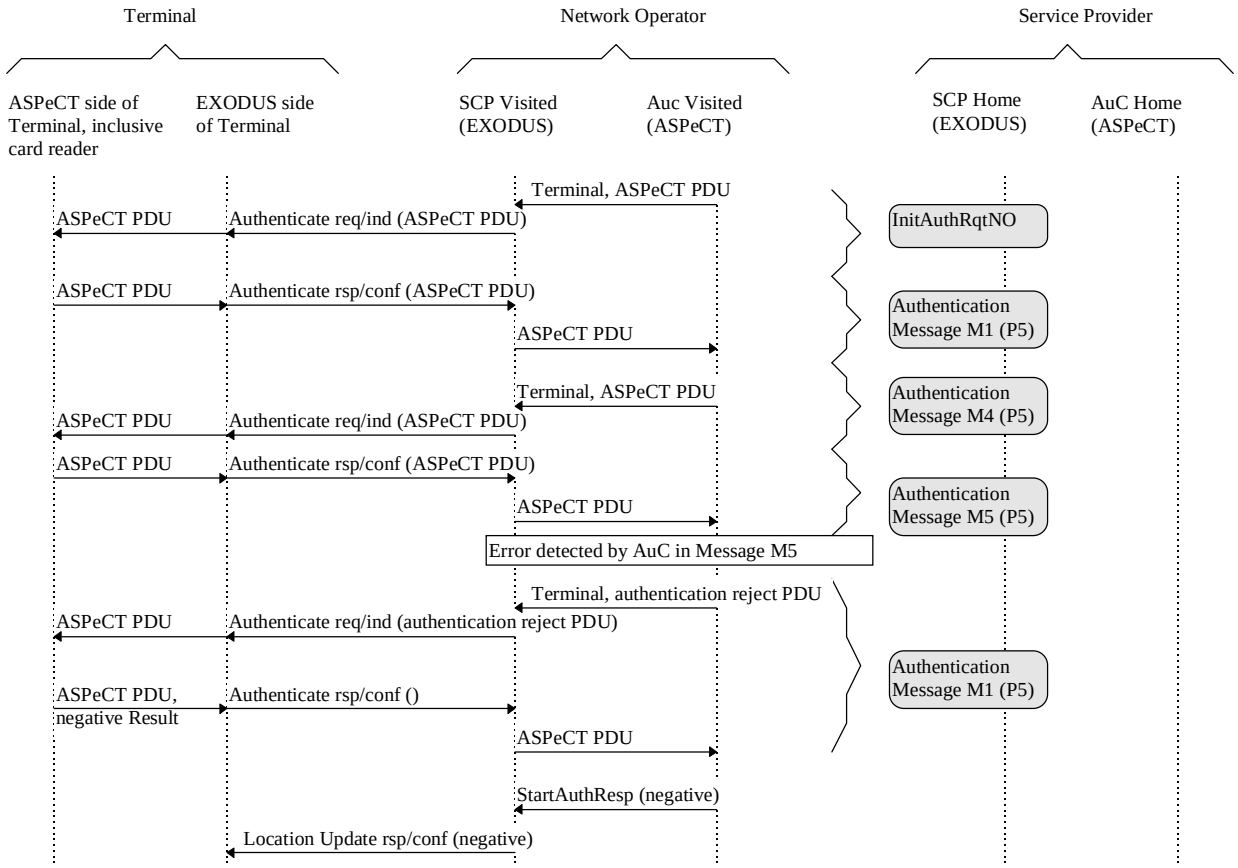

Figure 2.1.21 - Error in protocol detected by AuC

2.1.2.1.3.7 Overview of ASPeCT messages and EXODUS primitives

The following table lists the ASPeCT messages and the EXODUS primitives that will carry them.

ASPeCT Message	EXODUS primitive
InitAuthRqtNO	Authenticate req/ind
InitAuthRqt	Authenticate rsp/conf
CapsRqt	HandleInformationRequest req/ind
CapsInfo	HandleInformationRequest rsp/conf
PresAuthMech	Authenticate req/ind
Authentication Message M1	Authenticate rsp/conf
Authentication Message M2 (optional)	HandleInformationRequest req/ind
Authentication Message M3 (optional)	HandleInformationRequest rsp/conf
Authentication Message M4	Authenticate req/ind
Authentication Message M5	Authenticate rsp/conf
Authentication Reject	HandleInformationRequest rsp/conf
	Authenticate rsp/conf

Table 2.1.1 - Mapping of ASPeCT Messages to EXODUS primitives

2.1.2.1.4 Trial Scenarios

x = Test this item

o = Don't test this item

✓ = Test was successful

💣 = Test was unsuccessful

The purpose of the authentication trial is to show that the proposed security mechanisms work in an experimental UMTS environment and to measure network performance and quality of service as perceived by the users.

Different cases are being tested:

1. Multi-Media-Terminal versus DECT-Terminal
2. Intra-Network versus Inter-Network
3. Siemens versus the Royal Holloway University of London (Rhul) Protocol.

The Multi-Media-Terminal is based upon a PC which is in fact an EXODUS fixed broadband terminal. The DECT terminal is based on a EXODUS DECT terminal, which represents a mobile connection.

Intra-Network means, that the testing environment is restricted to the Swiss-National-Host-Platform. No international connections are possible. Inter-Network means, that the testing environment is not restricted to the Swiss-National-Host-Platform only. Registrations with the Italian-National-Host-Platform will be performed too.

ASPeCT implements two authentication mechanisms:

- the Siemens protocol B provides signalling between the Network Operator and the Service Provider;
- the Rhul protocol interrogates the Service Provider for new registrations.

The two protocol types that are being tested are the Siemens protocol and the Rhul one. The Siemens proposal for authentication defines a protocol A, B and C. Protocol A is the case where user and Network Operator share each others public key. In protocol B and C, certificates on each others public keys are exchanged. In the ASPeCT trial, protocol A is used for current registrations and protocol B is used for new registrations.

The authentication trial will be executed on the EXODUS Swiss National Host Platform in Basel by Jo Goedermans and Emmanuel Lachat from Swisscom.

2.1.2.1.4.1 Multi-Media-Terminal

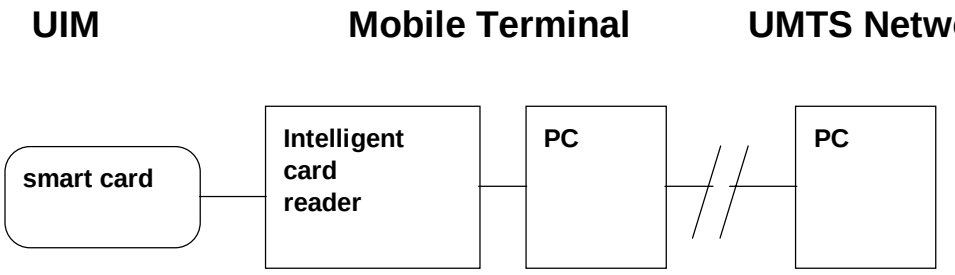

Figure 2.1.22 - Configuration with Multi-Media Terminal

2.1.2.1.4.1.1 INTRA-Network

2.1.2.1.4.1.1.1 Siemens Protocol

N°	Task + Description	1st. Reg.	2nd Reg.	nth Reg.	Expected Results
1.	No valuable registering information is available. Register with faulty PIN.	x	o	o	Registering may not be possible. The terminal should show an error message.
2.	Register normal 1st time.	x	x	x	Registering must be successful.
3.	After a normal successful registering, register with faulty PIN.	x	o	o	Registering may not be possible. The terminal should show an error message.
4.	Try to register with a faulty card.	x	o	o	Registering may not be possible. The terminal should show an error message.
5.	Take-out card during registration.	x	o	o	Registering may not be possible. The terminal should show an error message and should ask the user to register again.
6.	During Registration Stop AuC (only task- or SW-Stop).	x	o	o	Registering may not be successful. The terminal should show an error message and should ask the user to register again.
7.	AuC not connected (cut communication connection before registering). Try to register to SCP.	x	o	o	Registering may not be successful. The terminal should show an error message and should ask the user to register again.
8.	Cut-off communication link during registering.	x	o	o	Registering may not be successful. The terminal should show an error message and should ask the user to register again.
9.	Cut-off AuC Power (230VAC), during registration.	x	o	o	Registering may not be successful. The terminal should show an error message and should ask the user to register again. The process should remember what happened after starting-up.

2.1.2.1.4.1.1.2 Rhul-Protocol (Royal Holloway University of London)

N°	Task + Description	1st. Reg.	2nd Reg.	nth Reg.	Expected Results
1.	No valuable registering information is available. Register with faulty PIN.	x	o	o	Registering may not be possible. The terminal should show an error message.
2.	Register normal 1st time.	x	x	x	Registering must be successful.
3.	After a normal successful registering, register with faulty PIN.	x	o	o	Registering may not be possible. The terminal should show an error message.
4.	Try to register with a faulty card.	x	o	o	Registering may not be possible. The terminal should show an error message.
5.	Take-out card during registration.	x	o	o	Registering may not be possible. The terminal should show an error message and should ask the user to register again.
6.	During Registration Stop AuC (only task- or SW-Stop).	x	o	o	Registering may not be successful. The terminal should show an error message and should ask the user to register again.
7.	AuC not connected (cut communication connection before registering). Try to register to SCP.	x	o	o	Registering may not be successful. The terminal should show an error message and should ask the user to register again.
8.	Cut-off communication link during registering.	x	o	o	Registering may not be successful. The terminal should show an error to register message and should ask the user again.
9.	Cut-off AuC Power (230VAC), during registration.	x	o	o	Registering may not be successful. The terminal should show an error message and should ask the user to register again. The process should remember what happened after starting-up.

2.1.2.1.4.1.2 INTER-Network

2.1.2.1.4.1.2.1 Siemens Protocol

N°	Task + Description	1st. Reg.	2nd Reg.	nth Reg.	Expected Results
1.	No valuable registering information is available. Register with faulty PIN.	x	o	o	Registering may not be possible. The terminal should show an error message.
2.	Register normal 1st time.	x	x	x	Registering must be successful.
3.	After a normal successful registering, register with faulty PIN.	x	o	o	Registering may not be possible. The terminal should show an error message.
4.	Try to register with a faulty card.	x	o	o	Registering may not be possible. The terminal should show an error message.
5.	Take-out card during registration.	x	o	o	Registering may not be possible. The terminal should show an error message and should ask the user to register again.
6.	During Registration Stop AuC (only task- or SW-Stop)	x	o	o	Registering may not be successful. The terminal should show an error message and should ask the user to register again.
7.	AuC not connected (cut communication connection before registering). Try to register to SCP.	x	o	o	Registering may not be successful. The terminal should show an error message and should ask the user to register again.
8.	Cut-off communication link during registering.	x	o	o	Registering may not be successful. The terminal should show an error message and should ask the user to register again.
9.	Cut-off AuC Power (230VAC), during registration.	x	o	o	Registering may not be successful. The terminal should show an error message and should ask the user to register again. The process should remember what happened after starting-up.

2.1.2.1.4.1.2.2 Rhul-Protocol

N°	Task + Description	1st. Reg.	2nd Reg.	nth Reg.	Expected Results
1.	No valuable registering information is available. Register with faulty PIN.	x	o	o	Registering may not be possible. The terminal should show an error message.
2.	Register normal 1st time.	x	x	x	Registering must be successful.
3.	After a normal successful registering, register with faulty PIN.	x	o	o	Registering may not be possible. The terminal should show an error message.
4.	Try to register with a faulty card.	x	o	o	Registering may not be possible. The terminal should show an error message.
5.	Take-out card during registration.	x	o	o	Registering may not be possible. The terminal should show an error message and should ask the user to register again.
6.	During Registration Stop AuC (only task- or SW-Stop)	x	o	o	Registering may not be successful. The terminal should show an error message and should ask the user to register again.
7.	AuC not connected (cut communication connection before registering). Try to register to SCP.	x	o	o	Registering may not be successful. The terminal should show an error message and should ask the user to register again.
8.	Cut-off communication link during registering.	x	o	o	Registering may not be successful. The terminal should show an error message and should ask the user to register again.
9.	Cut-off AuC Power (230VAC), during registration.	x	o	o	Registering may not be successful. The terminal should show an error message and should ask the user to register again. The process should remember what happened after starting-up.

2.1.2.1.4.2 DECT-Terminal

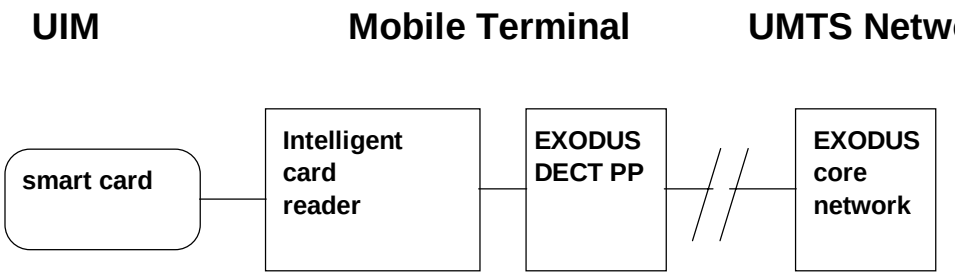

Figure 2.1.23 - Configuration with DECT Terminal

2.1.2.1.4.2.1 INTRA-Network

2.1.2.1.4.2.1.1 Siemens Protocol

N°	Task + Description	1st. Reg.	2nd Reg.	nth Reg.	Expected Results
1.	No valuable registering information is available. Register with faulty PIN.	x	o	o	Registering may not be possible. The terminal should show an error message.
2.	Register normal 1st time.	x	x	x	Registering must be successful.
3.	After a normal successful registering, register with faulty PIN.	x	o	o	Registering may not be possible. The terminal should show an error message.
4.	Try to register with a faulty card.	x	o	o	Registering may not be possible. The terminal should show an error message.
5.	Take-out card during registration.	x	o	o	Registering may not be possible. The terminal should show an error message and should ask the user to register again.
6.	During Registration Stop AuC (only task- or SW-Stop).	x	o	o	Registering may not be successful. The terminal should show an error message and should ask the user to register again.
7.	AuC not connected (cut communication connection before registering). Try to register to SCP.	x	o	o	Registering may not be successful. The terminal should show an error message and should ask the user to register again.
8.	Cut-off communication link during registering.	x	o	o	Registering may not be successful. The terminal should show an error message and should ask the user to register again.
9.	Cut-off AuC Power (230VAC), during registration.	x	o	o	Registering may not be successful. The terminal should show an error message and should ask the user to register again. The process should remember what happened after starting-up.

2.1.2.1.4.2.1.2 Rhul-Protocol

N°	Task + Description	1st. Reg.	2nd Reg.	nth Reg.	Expected Results
1.	No valuable registering information is available. Register with faulty PIN.	x	o	o	Registering may not be possible. The terminal should show an error message.
2.	Register normal 1st time.	x	x	x	Registering must be successful.
3.	After a normal successful registering, register with faulty PIN.	x	o	o	Registering may not be possible. The terminal should show an error message.
4.	Try to register with a faulty card.	x	o	o	Registering may not be possible. The terminal should show an error message.
5.	Take-out card during registration.	x	o	o	Registering may not be possible. The terminal should show an error message and should ask the user to register again.
6.	During Registration Stop AuC (only task- or SW-Stop).	x	o	o	Registering may not be successful. The terminal should show an error message and should ask the user to register again.
7.	AuC not connected (cut communication connection before registering). Try to register to SCP.	x	o	o	Registering may not be successful. The terminal should show an error message and should ask the user to register again.
8.	Cut-off communication link during registering.	x	o	o	Registering may not be successful. The terminal should show an error message and should ask the user to register again.
9.	Cut-off AuC Power (230VAC), during registration.	x	o	o	Registering may not be successful. The terminal should show an error message and should ask the user to register again. The process should remember what happened after starting-up.

2.1.2.1.4.2.2 INTER-Network

2.1.2.1.4.2.2.1 Siemens Protocol

N°	Task + Description	1st. Reg.	2nd Reg.	nth Reg.	Expected Results
1.	No valuable registering information is available. Register with faulty PIN.	x	o	o	Registering may not be possible. The terminal should show an error message.
2.	Register normal 1st time.	x	x	x	Registering must be successful.
3.	After a normal successful registering, register with faulty PIN.	x	o	o	Registering may not be possible. The terminal should show an error message.
4.	Try to register with a faulty card.	x	o	o	Registering may not be possible. The terminal should show an error message.
5.	Take-out card during registration.	x	o	o	Registering may not be possible. The terminal should show an error message and should ask the user to register again.
6.	During Registration Stop AuC (only task- or SW-Stop).	x	o	o	Registering may not be successful. The terminal should show an error message and should ask the user to register again.
7.	AuC not connected (cut communication connection before registering). Try to register to SCP.	x	o	o	Registering may not be successful. The terminal should show an error message and should ask the user to register again.
8.	Cut-off communication link during registering.	x	o	o	Registering may not be successful. The terminal should show an error message and should ask the user to register again.
9.	Cut-off AuC Power (230VAC), during registration.	x	o	o	Registering may not be successful. The terminal should show an error message and should ask the user to register again. The process should remember what happened after starting-up.

2.1.2.1.4.2.2.2 Rhul-Protocol

N°	Task + Description	1st. Reg.	2nd Reg.	nth Reg.	Expected Results
1.	No valuable registering information is available. Register with faulty PIN.	x	o	o	Registering may not be possible. The terminal should show an error message.
2.	Register normal 1st time.	x	x	x	Registering must be successful.
3.	After a normal successful registering, register with faulty PIN.	x	o	o	Registering may not be possible. The terminal should show an error message.
4.	Try to register with a faulty card.	x	o	o	Registering may not be possible. The terminal should show an error message.
5.	Take-out card during registration.	x	o	o	Registering may not be possible. The terminal should show an error message and should ask the user to register again.
6.	During Registration Stop AuC (only task- or SW-Stop)	x	o	o	Registering may not be successful. The terminal should show an error message and should ask the user to register again.
7.	AuC not connected (cut communication connection before registering). Try to register to SCP.	x	o	o	Registering may not be successful. The terminal should show an error message and should ask the user to register again.
8.	Cut-off communication link during registering.	x	o	o	Registering may not be successful. The terminal should show an error message and should ask the user to register again.
9.	Cut-off AuC Power (230VAC), during registration	x	o	o	Registering may not be successful. The terminal should show an error message and should ask the user to register again. The process should remember what happened after starting-up.

2.1.2.2 Initialisation of trial

This chapter gives the initialisation and personalisation data as will be used during the trial.

2.1.2.2.1 General initialisation information

The trial configuration will have a Swiss groups of entities (with a Network Operator, a Service Provider, a certification authority and 20 users) and an Italian one (with a Network Operator, a Service Provider, a certification authority and 10 users).

The actual trial data is included in Annex 1.

2.1.2.2.2 Personalisation of smart cards

During personalisation each smart card is loaded with both general data for the UMTS authentication, for example, the parameters which describe the elliptic curve cryptosystem being used, and also the EXODUS

data which is particular to the specific card. This latter data comprises the IMUI, the Authentication Capability Class, the user's cryptographic keys and the PIN (See Annex 1).

2.1.2.2.3 Initialisation for the AuC

The Authentication Centre uses the *c:\windows\asptrial.ini* file to retrieve initialisation information from. The file contains the following data:

- authentication mechanisms the AuC is able to handle,
- level of detail for logging,
- identity of the associated Service Provider,
- list of the recognised certification authorities and their public key,
- identity of the associated Network Operator,
- public and secret key of this Network Operator,
- certificate of this Network Operator, for each of the certification authorities,
- for each user: IMUI, public key, TMUIs, TMUIn, Knu and Ksu.

At startup, the SP and NO identity, the handled authentication mechanisms and the logging detail, can be changed by the AuC operator.

2.1.2.3 User Documentation and Training

The Authentication trial will consist of a number of users making calls across the EXODUS platform and evaluating the performance of the authentication framework as an integral part of the call set up procedure. However, an evaluation of this type clearly needs a structure to allow results to be obtained that can be quantified. To this end, as part of the trial, each user will be given a document outlining what procedure should be followed, and how the results of each action should be reported. This will correspond to a list of instructions, with space after each instruction for a structured response to be given. The results thus obtained will then be processed to assess the authentication by allowing a comparison to be made between such things as authenticated vs. non-authenticated call set up times. The full range of tested criteria will be defined as part of the trial documentation.

To prepare the users for the trial, each user will be given an introductory document prior to the trial. This will state the background and the scope and purpose of the trial, and how the results will be used. This will provide the context of the trial required by the users.

After the trial, the results obtained will be made available to each user for their interest. The final results will be included in the final ASPeCT project report.

2.1.2.4 Trial Evaluation

To evaluate the trial, two assessment criteria are used, namely *quality* and *performance*. Neither criterion on its own is sufficient to evaluate the system, as a successful system must both be of a usable quality and adequate performance.

The Quality of the system is more related to the users view of the system, whilst the Performance of the system is more related to the network providers' view. However, it is clear that there is some degree of overlap in these definitions.

To assess the technical feasibility of the trialled system, the performance of the system will be measured directly from the system's log files as defined in section 2.1.2.4.1. To asses the quality of the system, the response of the users to various actions will be recorded and assessed as defined in section 2.1.2.4.2.

2.1.2.4.1 Evaluation of Technical Feasibility

To evaluate the technical feasibility of the solution, performance measurements will be made as part of the authentication user trial. This data will be obtained in the form the log files produced by the AuC, SCP and Terminal.

This data is recorded using a simple tag system (as defined in [V100]), which should make it a fairly straightforward matter to deduce the various timings for each part of the call set up process.

These timings will then be compared against the gross timings defined within the GSM network to assess whether an adequate performance is achieved, and to determine what overhead is added by the authentication framework.

In addition to the performance measurements made, the approach taken of using the authentication framework will be assessed to see if it provides the functionality required to satisfy the desired features of an authentication process as defined at the start of the project.

2.1.2.4.2 Evaluation of User Acceptability

The trial shall assess the Quality and hence the user acceptability of the system in three ways, each representing a separate section of the user trial documentation.

The first section shall be used to determine the user's perceived response of the system to a series of individual actions, and the user shall record these using a five level scale of satisfaction: {Bad, Poor, Fair, Good, Excellent}. Once these are averaged for all users, an average level of response will be attained for each action.

The second section will record the overall assessment of the system by each user and will use a group of more general questions at the end of the list of actions. These will again use this scale of satisfaction, and will ask questions about the overall usability and quality of the system. Once these are averaged for all users, an overall level of satisfaction will be recorded.

Finally, a third comments section shall allow feedback on issues not covered specifically under the previous two sections, and allow the users to input more individual views. These will be analysed to see if there are any common responses, and these will also be recorded.

2.1.3 Demonstration Configuration

2.1.3.1 PC - based configuration

The authentication demonstration configuration consists of:

- the AuC, which is equal to the trial version;
- a User emulator, which is basically a shell written around the ASPeCT terminal software, allowing multiple users to be emulated;
- an SCP emulator, which has also a built-in User emulator.

These can be combined in a number of ways. The only requirement is that every AuC has to be connected to only one SCP and vice-versa.

The minimal configuration uses the AuC and the SCP emulator, both running on one PC. This is the easiest way to demonstrate the authentication, but it does not allow home Service Provider interrogation.

The maximal configuration is as follows: an AuC and a User emulator are connected to a SCP emulator, which is then connected to another SCP emulator, with an attached AuC and User emulator. The six participating entities can be distributed over up to 6 PCs, providing that a TCP/IP connection is possible between them.

A typical test configuration for the AuC is to attach it to the SCP emulator. The SCP can be tested by connecting it both to an AuC and a User emulator (both can run on the same PC). For demonstration purposes, the configuration of Figure 2.1.24 is most likely to be used. Two interconnected PCs are used, and each PC runs an AuC, a User emulator and an SCP emulator.

A tracer program allows to check the authentication messages while the programs are running. The programs themselves also show what they are currently doing.

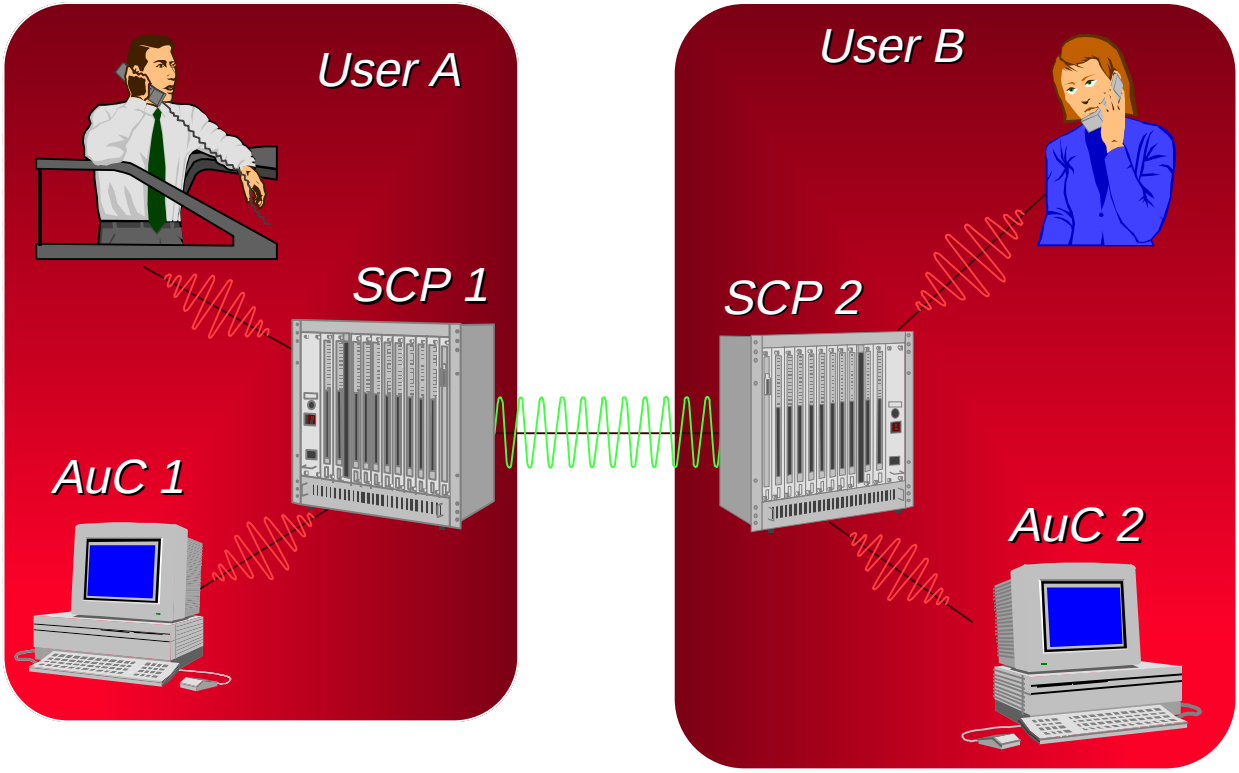

Figure 2.1.24 - Authentication Demonstration Configuration

2.1.3.2 Smart card functionality

In addition to the UIM functionality on the smart card there will also be other applications present to demonstrate the possibilities and opportunities offered by multi-application smart cards. The following two applications will be demonstrated:

- **GSM SIM Application**

  This will be a standard GSM SIM and will show the possibility of having a combined U(S)IM/SIM during the migration phase as UMTS gradually subsumes GSM. The SIM application will be a completely independent application on the card and it will be possible to plug the card into a GSM mobile phone and make a call using the Vodafone network. The same card could then be used in the ASPeCT authentication demonstration or to perform authentication in the EXODUS trial environment.

- **Visa Cash – Public Key Electronic Purse**

This will be a real world demonstration card for making electronic payment using a public key cryptography based on the prEN1546 architecture [PrEN96]. The application will be based on an existing standalone Visa Cash implementation but, since it is only a demonstration card will contain only test keys. A separate demonstration program will be used to show the functionality of the Visa Cash card. This application will show the possibility of using established electronic payment systems to pay for services over telecommunication links.

The Visa Cash Public Key Purse is an EMV compatible smart card application which supports Load, Purchase, Incremental Purchase and Update transactions. The Purchase transactions are secured using RSA digital signatures and can thus be verified by an off-line purchase terminal. The Load and Update commands are secured using MACs calculated using single length DES session keys, the session key is derived from a double length DES key. The use of symmetric cryptography for these commands means that they must either be performed using an on-line connection to a secure host or in a secure environment.

Note: The Visa Cash application itself will be slightly modified to prevent its use in a real Visa Cash terminal even if the correct keys were available.

It should be noted that only one of these two additional applications will be present on a card at any time.

2.1.4 Realisation

2.1.4.1 Graphical User Interface

At the initialization of the application a message box appears prompting the user to establish an SCP Connection. The main window is activated as soon as a new SCP connection is established and a dialog box is displayed in which the user can redefine the Network Operator ID, the Service Provider ID, the Authentication Mechanisms known by the Network Operator and the level of detail of the traced information (Figure 2.1.25).

The GUI main menu bar comprises three items, namely (see Figure 2.1.25):

- **Application**
- **Entity** and
- **Help**

The **Application** item enables the re-initialization of the application database, the initialization of the Tracer, the termination of the application. It also gives some overall information about the application database. The **Entity** item allows the entity resetting and provides specific information about the already created entities. Finally, the **Help** item will include information regarding the GUI components. More detailed information regarding the main window and the menu items is given below.

2.1.4.1.1 The Main Window

The main window (Figure 2.1.26) displays:

- the Service Provider,
- the Network Operator,
- two information windows concerning the above entities, and
- a status line.

By clicking the left mouse button on one of the two bitmaps a new message box appears displaying information about the corresponding entity.

The two information windows enable the trial process observation. Each time a new authentication request is made, a new authentication process is initiated and assigned an FSM ID. Two fields are updated displaying

information about the Transaction ID and the state of the corresponding authentication process. Meanwhile, each status button can activate a Status window for the respective FSM ID displaying information such as certificates, keys signatures etc.

Finally, at the bottom of the main window there is a status line indicating the menu item that was recently selected by the user.

2.1.4.1.2 The Application Part

The **Application** pull-down menu comprises four items, namely (Figure 2.1.17):

- *New*
- *Status*
- *Trace*
- *Exit*

By clicking the left mouse button on the *New* items, the application database is re-initialised and the TCP/IP connection is reset. Upon selecting the *Status* item a message box appears displaying full application database information. The **Application** pull-down menu also initiates the *Tracer* option. The Tracer displays the messages exchange process as well as the status of each created entity via a new window appearing in the right side of the screen. When the *Tracer* menu item is selected a check mark is displayed next to it in order to indicate that the Tracer is activated. Finally, the *Exit* option terminates the application.

2.1.4.1.3 The Entity Part

The **Entity** Part (Figure 2.1.28) comprises two menu items, namely

- *Reset*, and
- *Status*

The *Reset* menu item activates a new dialog box which includes a list box with all the created instances of the NO FSM. In order to reset one of these, the user has to mark it in the list box and click the *Reset* button. By clicking the *Status* menu item, a list box with all the created entities (the main NO, the main SP and all created instances of the NO FSM) appears and the user is able to select the entity whose status he chooses to review. By then clicking the *Display Status* button, a new dialog box appears providing information on the corresponding entity. This is an alternative way to view the entities status windows instead of clicking on the bitmaps and the status buttons from the main window. An example status window is shown in Figure 2.1.29.

2.1.4.1.4 The Help Part

The **Help** pull-down menu options will enable the user to search through the *Contents*, to be informed on *How to Use Help* and be informed *About* this trial. The contents of these options are not incorporated in this version.

Figure 2.1.25 - The initial GUI Menu Screen

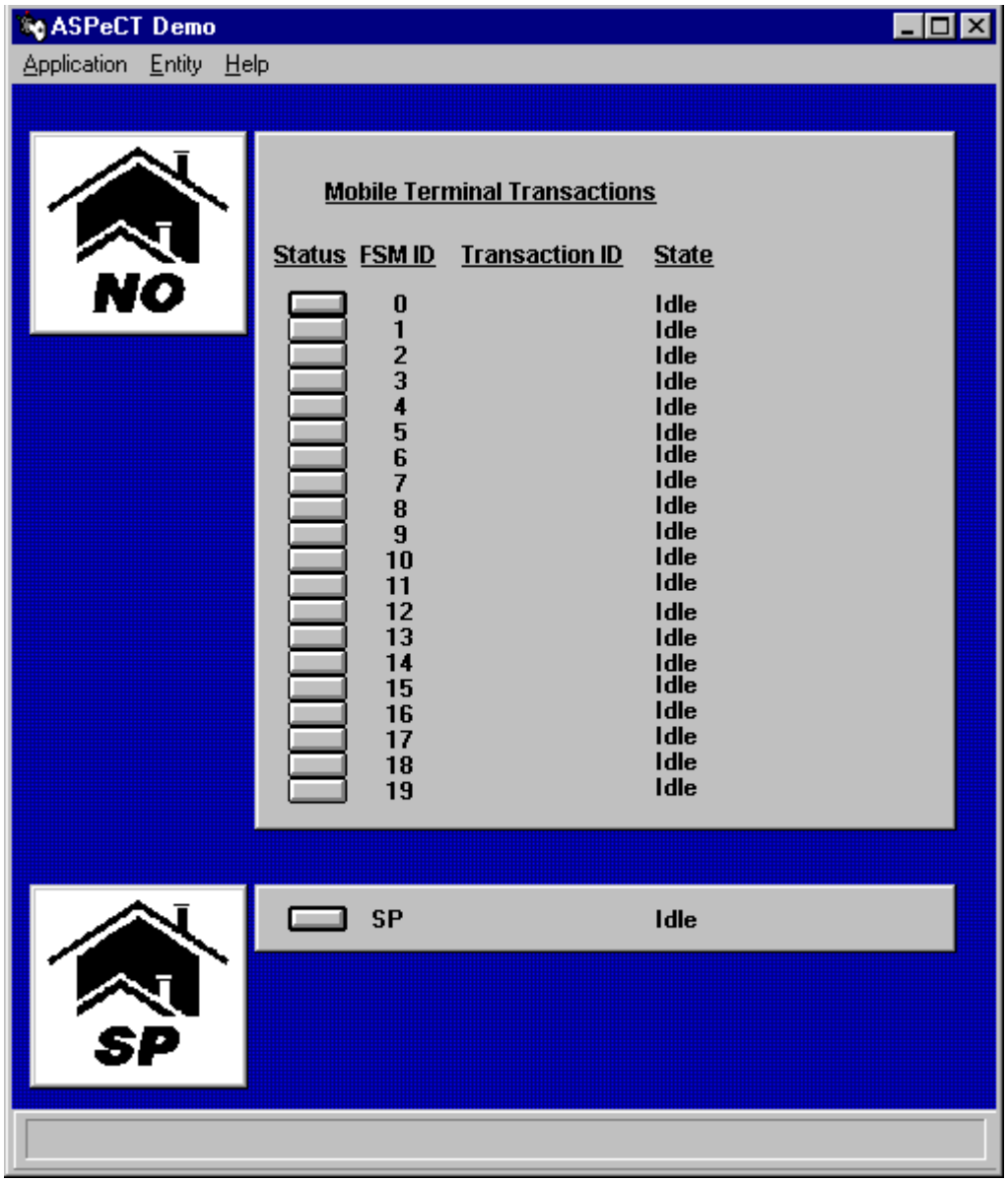

Figure 2.1.26 - The Main Window

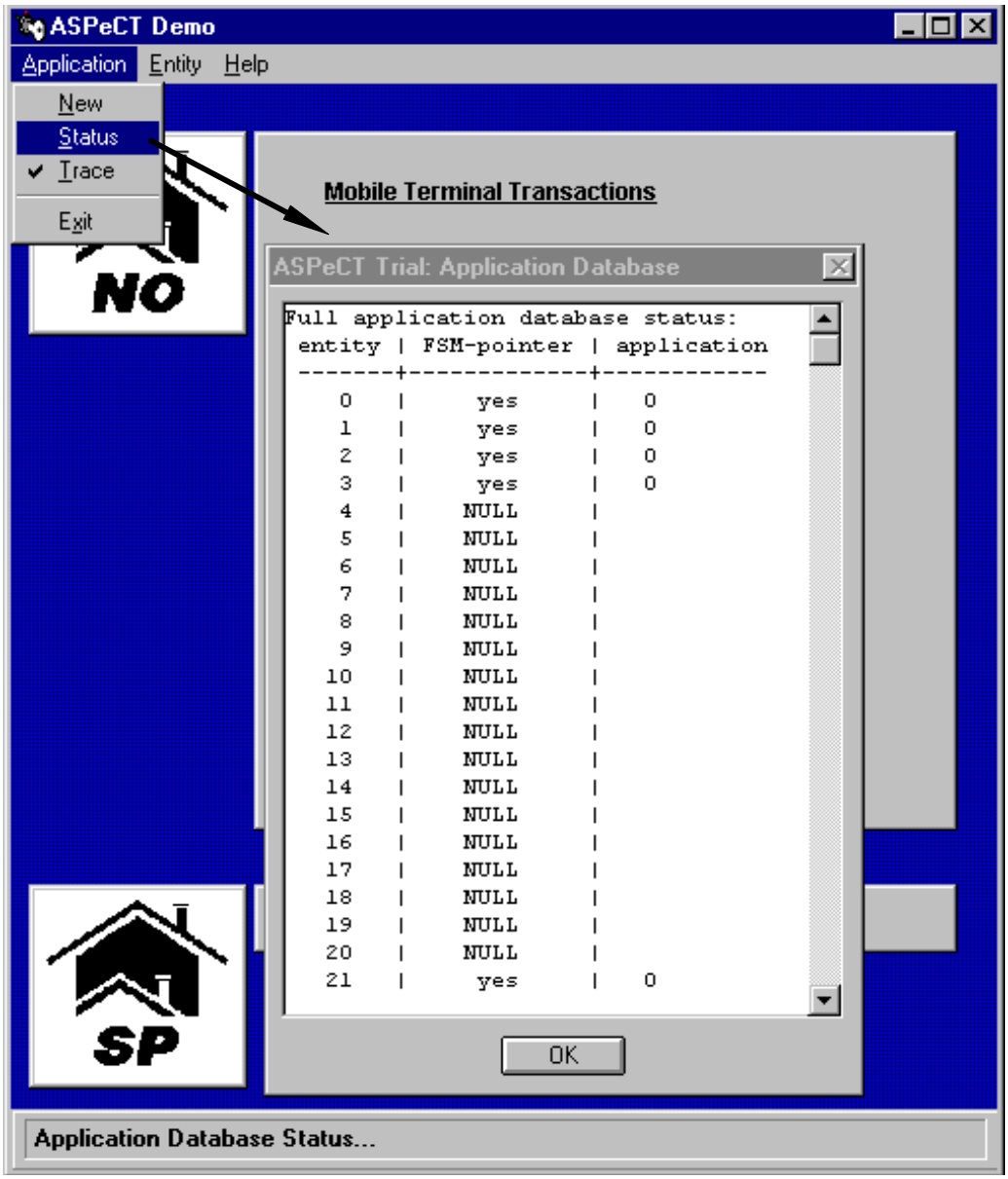

Figure 2.1.27 - The Application *pull-down menu*

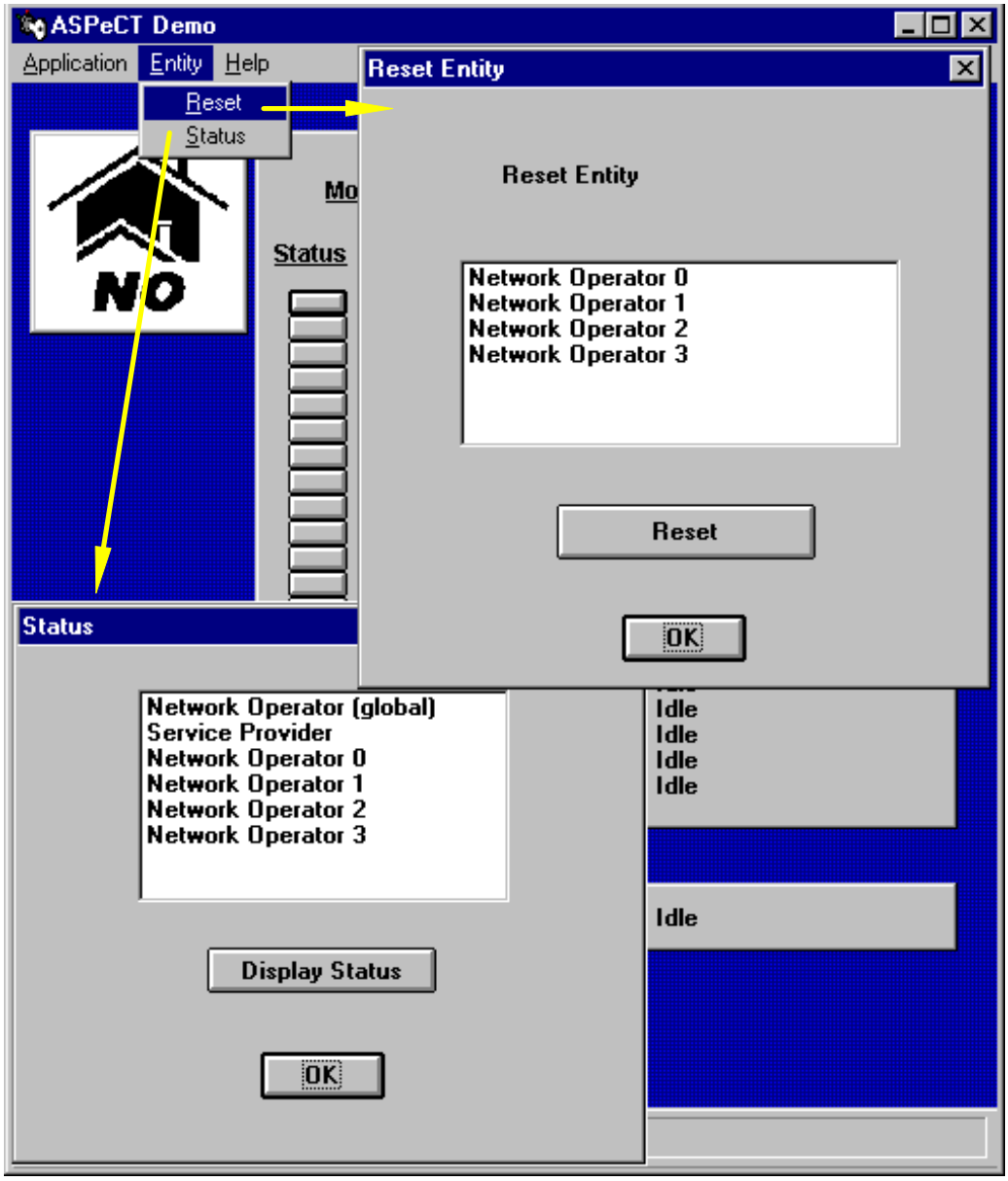

Figure 2.1.28 - The Entity *pull-down menu*

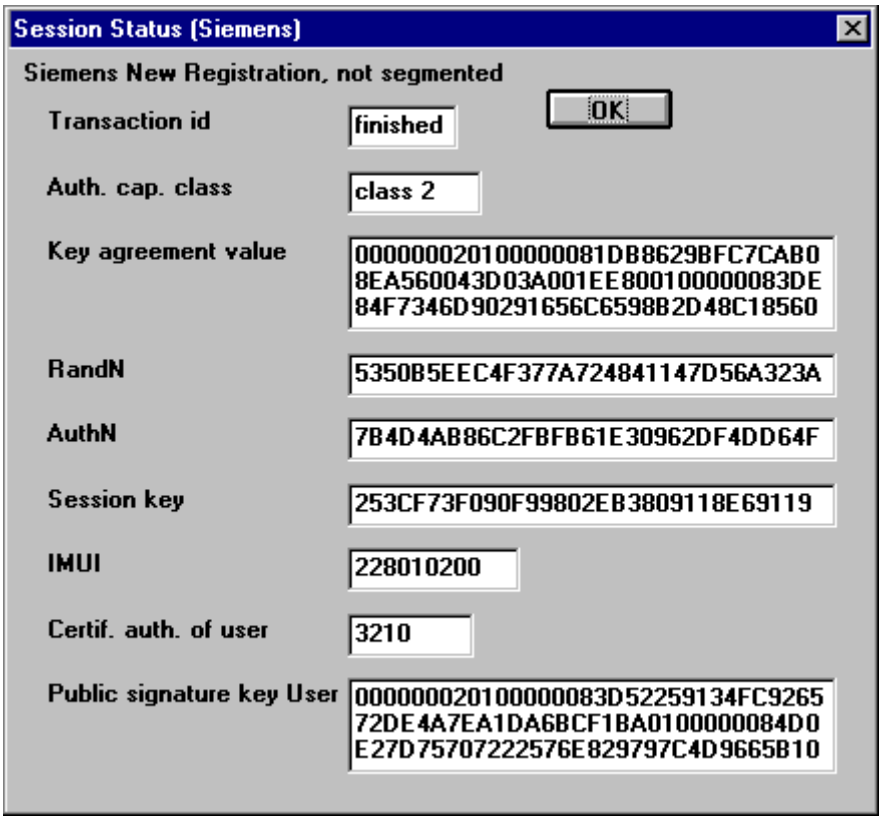

Figure 2.1.29 - An example Status *window*

2.1.4.2 Software architecture

2.1.4.2.1 Smart card

The interface between the smart card reader and the PC side of the ASPeCT terminal software is implemented by the COLA dynamic link library. Note that this interfaces with, but is not the same as the CoLa security subsystem shown in Figure 2.1.30.

This library presents a clean interface with application rather than implementation related functions. These functions are translated into combinations of card and card reader commands as appropriate. The interface hides the true card interface from the ASPeCT PC application and allows different types of smart card implementations to be used completely transparently to the application software. For example, the multi-application UIM used in the trials together with EXODUS uses a different communication protocol to the migratory UIM / GSM SIM. Furthermore, the COLA DLL also offers and emulation mode where it is possible to run the program without using a UIM or card reader. In this case any data persistence is implemented using an initialisation file on the PC. Finally the COLA DLL supports the optional logging of its operation to a log file. This is useful for both statistical and debugging purposes.

The following table shows how the functionality provided by the COLA DLL is translated into card or card reader commands. Note that the functionality required specifically for the EXODUS project uses only standard ISO/IEC 7816-4 [ISO95] commands whilst the UMTS authentication uses commands specially developed for ASPeCT. The table does not show the commands used to log data or to display informational text on the card reader's display.

	Reader Commands					Card Commands											
	InitialiseSerial	Close Serial	Initialise ICC	Eject	Status	Select	Verify PIN	Read Binary	Write Binary	Read Record	Update Record	Get Data	Put Data	Read IMUI	Read Certificate	Get Challenge	Mutual Auth
cola_OpenReader	•																
cola_OpenUIM	•		•			•											
cola_CardPresent					•	•											
cola_CloseUIM		•		•													
cola_SelectApplication						•											
cola_VerifyUser						•	•										
cola_ReadAuthCapClass								•									
cola_ReadServProvID								•									
cola_ReadCertAuthID								•									
cola_ReadIMUI														•			
cola_VerifyNOCert									•								
cola_GetChallenge																•	
cola_MutualAuthenticate														•	•		•
cola_ReadExodusIMUN										•							
cola_WriteExodusIMUN											•						
cola_ReadExodusTMUI								•									
cola_WriteExodusTMUI									•								
cola_ReadExodusLAI_FPAI								•									
cola_WriteExodusLAI_FPAI									•								
cola_ReadExodusLID								•									
cola_WriteExodusLID									•								
cola_ReadExodusCPN								•									
cola_WriteExodusCPN									•								
cola_ReadExodusKs								•									
cola_WriteVoicePrint											•						

Table 2.1.2 - COLA DLL functionality translated into card and card reader commands

The COLA DLL actually uses another DLL, PC/CTI, to transfer the APDUs between the card and the PC.

The architecture of the UIM comprises several major components of which the most important are:

- Command Manager – this verifies the parameters of a command and then dispatches it to the appropriate handler for the particular command.
- File System Manager – The file system is created and initialised during the personalisation and contains all the card specific or potentially variable data. The File System Manager controls access to the file system but consults the Security Manager to confirm that such access is permitted. The File Manager is also responsible for ensuring that the data from one application is not accessed by another application on a multi-functional card.
- Security Manager – the security manager controls the access of an application to its files. Only if the appropriate security conditions have been met will the Security Manager allow access to the data on the card. The Security Manager is implemented as a finite state machine.

These components are implemented in the ROM code of the smart card. Additional features, or modifications, which are needed for the implementation of the ASPeCT UIM have been implemented in the EEPROM memory of the card.

2.1.4.2.1.1 The File System

The file systems on the multi-application UIM and the migratory UIM are different – this is because they have been implemented on different platforms. The platform for the multi-application UIM is specially designed as a general purpose operating system for public key applications (such as the UIM authentication application) whilst the migratory UIM platform is designed for GSM applications.

The following describes the file system as it is on the multi-application UIM. The concept for the migratory UIM is the same but there no ISFs or IPFs are available so that they must be implemented as normal files.

File	Type	File Identity	Short File ID	Description
DF_UMTS	DF	7F50		UMTS Application
EF_SPID	Binary	6F00	19	Service Provider ID
EF_AUCC	Binary	6F01	01	Authentication Capability Class
EF_IMUI	Binary	6F02	02	IMUI
EF_CERTU	Binary	6F05	05	User certificate
EF_CERTN	Binary	6F06	06	Network Operator certificate
EF_SECU	Binary	6F07	07	User's secret key
EF_IPF	IPF	6F80	1D	Internal Public key file 1: Certification Authority public key 2: Domain parameters of Elliptic Curve (*p*,*q*,*A*,*g*) 3: Network Operator public key
EF_CAID	Binary	0018	18	Certification Authority ID
EF_IMUN	Fixed Record	0011	11	EXODUS IMUNs 17 records of 10 bytes
EF_MISC	Binary	0012	12	EXODUS TMUI, LAI/FPAI, LI, CPN
EF_KS	Binary	001E	1E	Session Key, Ks
EF_TLV	Object	1000		Operating system Object file

Table 2.1.3 - File system on multi-application UIM

2.1.4.2.1.2 The Commands

The commands implemented by the UIM can be divided into two groups – those which are generic (compatible to ISO/IEC 7816-4 [ISO95]) and those which have been specially implemented for the UMTS application. This section briefly describes the commands.

Select File	Used to select UMTS application by selecting its file identifier
Verify	Authenticate the user by checking his PIN, optionally allow the user to change the PIN to a new value. Only after a successful verification will the Security Manager allow files on the UIM to be accessed

Read Binary	Read the content of a transparent file. The offset into the file and the short file identifier of the file are specified in the command
Read Record	Read the content of a record structured file. The record number and the short file identifier are specified with the command.
Get Challenge	This generates a random number RND_U and uses this to calculate the card's challenge using arithmetic in the Elliptic Curve group. This value is then returned by the card. The random is stored for later use.
Mutual Authenticate	This performs the bulk of the processing on the UIM. Given RND_N, $AUTH_N$ and Enc(K_S; Token1) the UIM calculates the corresponding session key K_S and verifies the RND_N and $AUTH_N$ values are consistent with the RND_U value generated previously. If successful the UIM returns Enc(K_S; Token2) and stores the session key K_S in EF_KS. The previously stored random RND_U is deleted irrespective of the success of the authentication. The Mutual Authentication command will only run successfully if the RND_U value has not been deleted.
Verify Certificate	This command verifies the data which has been previously written in EF_CERT. If this data represents a valid certificate then the encapsulated ID and public key are written to record 3 of the IPF.
Read IMUI	This allows the IMUI to be read out from the card. For the ASPeCT authentication the value is read out enciphered by the session key K_S. EXODUS required the session key to be read out in the clear so the command also supports this option.
Read Certificate	This allows the user certificate to be read out from the EF_CERT file enciphered by the session key K_S

2.1.4.2.2 ASPeCT terminal software

The ASPeCT side of the terminal application is presented as a dynamic link library (DLL). It has two main functions for the EXODUS side of the terminal: it handles the authentication messages in order to provide a session key, and it offers an interface to the EXODUS data on the smart card.

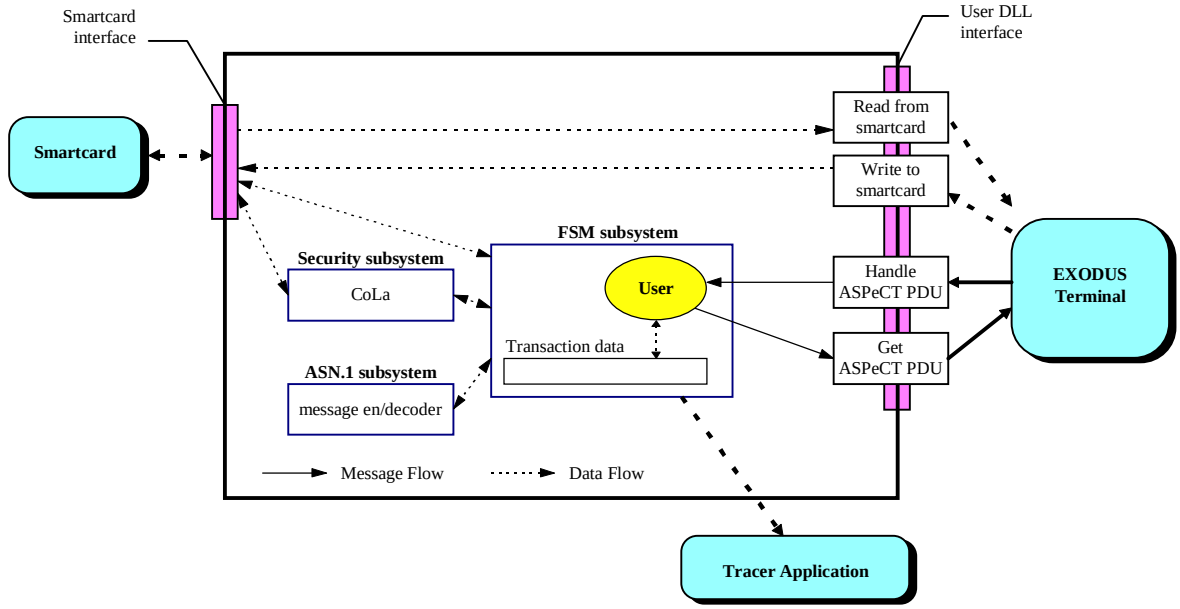

Figure 2.1.30 - ASPeCT terminal software

Read/Write requests for smart card data are mapped directly to corresponding smart card commands, and the response is returned to EXODUS.

Incoming ASPeCT messages are handed over to a *finite state machine* (FSM) *subsystem* representing the User side of authentication. To compute a response message, the FSM subsystem makes use of the security subsystem, the ASN.1 subsystem and the data on the smart card.

The *security system* is in fact nothing more than a conversion layer (CoLa) between the FSM subsystem and the cryptographic functions on the smart card.

Details about the negotiation and authentication messages can be displayed using a separate character-oriented *tracer* application. This information includes time stamps, encoded and decoded messages etc. The ASPeCT terminal software communicates with the tracer server process via a dynamic link library (DLL).

In order to do ASN.1 encoding and decoding, the FSM subsystem uses the functions of the *ASN.1 subsystem*. These functions have been generated from the ASN.1 definitions by use of “snacc” (Sample Neufeld Asn.1 to C/C++ Compiler).

2.1.4.2.3 AuC

The Authentication Centre consists of six large subsystems (see Figure 2.1.31). The two most important of them are the *finite state machines (FSM) subsystem*, containing the Network Operator and Service Provider entities, and the *communications subsystem*, which is responsible for the handling of the authentication messages. These messages are exchanged in transactions (a transaction consists of a group of messages, exchanged with the SCP, and related to a specific call).

The NO is able to handle a number of transactions in parallel, which is necessary because it has to remember its state between messages of the same transaction. The SP only handles one transaction at the time, which is sufficient as it is a stateless entity.

Via the message handler, NO and SP can exchange messages with each other (using a message queuing mechanism) and with the SCP (over TCP/IP). A process in the message handler continuously polls the queue and the TCP/IP connection, and routes messages on it to the correct destination.

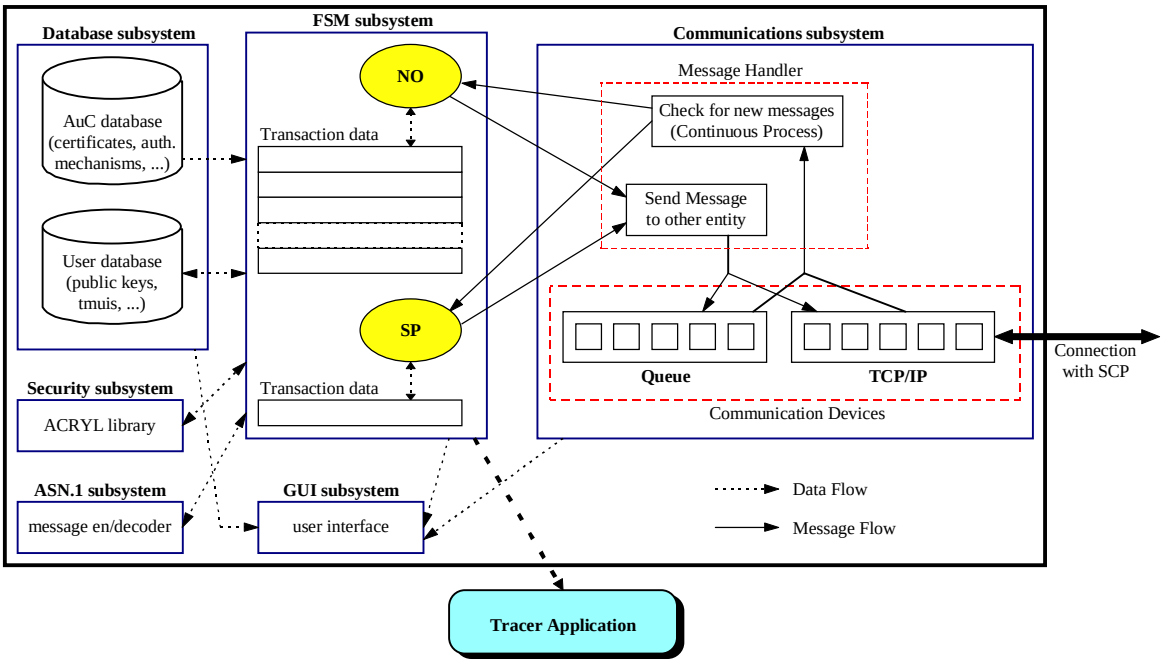

Figure 2.1.31 - Authentication Centre

The FSM subsystem makes use of information which is managed in the *database subsystem*. This contains persistent information such as NO and SP identity, NO certificates, NO keys, recognised authentication mechanisms (for the AuC itself), and public keys, tmuis, ksu (information about users).

The *security subsystem* helps the FSM subsystem to perform its authentication task. Security is built around the Advanced CRYptographic Library (ACRYL), which is provided by Siemens AG.

Details about the negotiation and authentication messages can be displayed in four levels of detail using a separate character-oriented *tracer* application. This information includes time stamps, encoded and decoded messages etc. The AuC communicates with the tracer server process via a dynamic link library (DLL).

In order to do ASN.1 encoding and decoding, the FSM subsystem uses the functions of the *ASN.1 subsystem*. These functions have been generated from the ASN.1 definitions by use of “snacc” (Sample Neufeld Asn.1 to C/C++ Compiler).

Information that relates to the current state of the entities and the transactions in the AuC is available via the GUI of the AuC. The GUI subsystem receives information from the database subsystem, the communications subsystem and the FSM subsystem. Part of the information is always visible, such as the identification and state of currently processed transactions. The rest of the information is available on request, by clicking on the NO, SP or transaction buttons.

The ASPeCT software is implemented as a 16-bit application, and will be running in a Windows 95 environment on the AuC PCs.

2.1.5 Plan

2.1.5.1 Milestones and Key Dates

Integration test:

Date	Place	Description
1-4 December	Coventry	SCP-AuC connection test

9-12 February	Bern	complete integration test
9-12 March	Basel	final integration test

Dedicated authentication test:

Date	**Place**	**Description**
23-24 March	Basel	Intra-network test
2-3 April	Basel	Inter-network test

2.1.5.2 Risks, Limitations and Dependencies

2.1.5.2.1 Concerning Project EXODUS

Authentication can be controlled in the SCP, the execution of the authentication protocol is dependent on the flags set within the SCP, under control of EXODUS. The EXODUS users have to use authenticated connections.

2.1.5.2.2 Arising from ASPeCT

None.

2.2 Secure Billing and TTP Trial

2.2.1 Technical Description

This section presents a high level technical description of the demonstrator implemented in the TTP and secure billing field trial.

2.2.1.1 Overview of architecture

The joint TTP and Secure Billing demonstrator (demo 2) consists of three entities each represented by a separate PC. The entities are:

- a mobile user;
- a value-added Service Provider (VASP);
- a trusted third party (TTP).

The service offered by the VASP is a WWW service which enables the user to retrieve value-added information using a WWW browser. In addition, there are processes running on both the user and the VASP PCs to execute the secure billing protocols. Furthermore, there is a process running on the TTP PC which acts as a certificate server. When the authentication between the user and the VASP takes place, the VASP retrieves certificates on-line from the TTP.

The three PCs communicate across the EXODUS ATM network using the protocols described in Section 2.2.1.5. For the communication between the user PC and the VASP PC, EXODUS provides a specialized driver to realise Windows Sockets over an ATM connection. For the communication between the VASP and the TTP, an InterWorking Unit (IWU) is provided by EXODUS

All the detailed information that might be of interest to an observer (e.g. details of protocol messages) is displayed using a character-orientated tracer running on each of the PCs.

Monitor points are provided in the code to enable performance measurements to be written to log files.

The ASPeCT software will be implemented as a 32-bit Windows application and will be run in a Windows NT 4.0 environment on the EXODUS terminal PCs.

2.2.1.2 Architecture of User Software

On the user PC, two main processes are running: the WWW browser, which is the client for the VAS offered by the VASP; and the secure billing application, which handles the payment. The WWW browser uses Windows Sockets for communications with the WWW server. One part of the secure billing software is a layer between the WWW browser and the Windows Sockets stack which intercepts all communication data. This interception software informs the secure billing application about selected events which occur during execution of the WWW service.

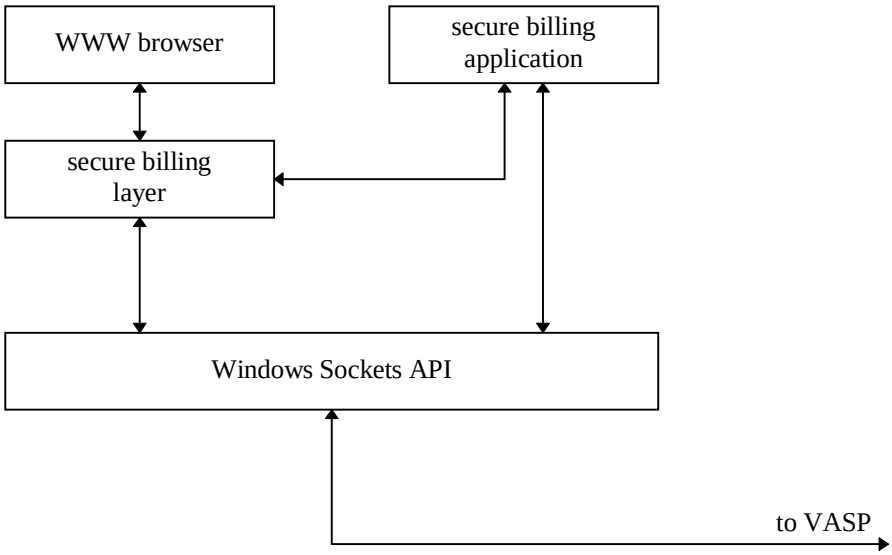

Figure 2.2.1 - Architecture of user software

2.2.1.3 Architecture of VASP Software

On the VASP PC, two main processes are running: the WWW server, which offers the VAS to clients; and the secure billing application, which handles the payment. The WWW server uses Windows Sockets for communications with the WWW clients. One part of the secure billing software is a layer between the WWW server and the Windows Sockets stack which intercepts all communication data. This interception software informs the secure billing application about selected events which occur during the execution of the WWW service.

Communication with the TTP application is also via Windows Sockets. The sequencing is simple, with a request to the TTP resulting in an associated response; there is no asynchronism.

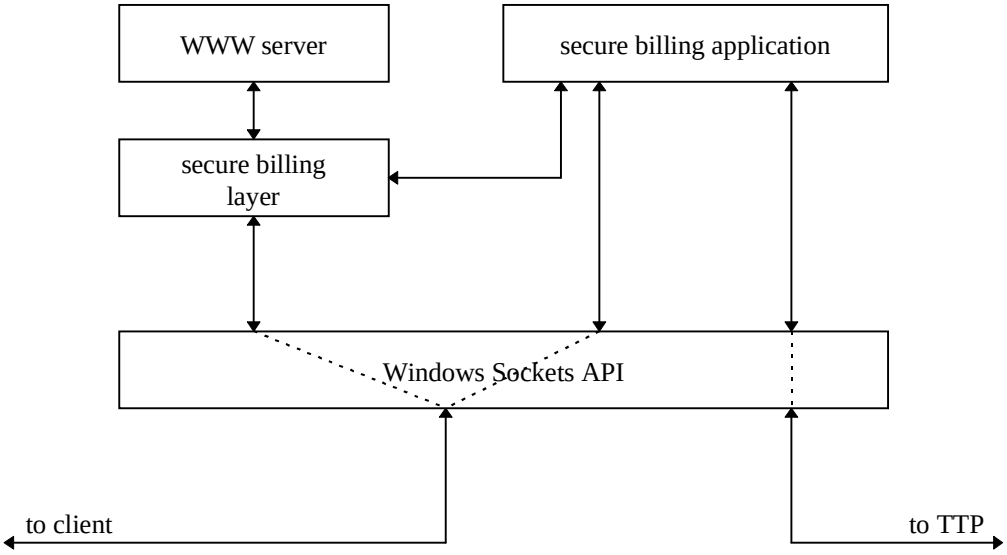

Figure 2.2.2 - Architecture of VASP software

2.2.1.4 Architecture of TTP Software

On the TTP PC, two main processes are running; these provide two basic services that would be offered by a more complete Trusted Third Party definition [D07]:

- the trusted third party application (TTPA), which provides signed, timestamped certificate information in response to requests in the protocol between the TTP and VASP;
- a certificate service application (CSA) which prepares and signs certificates that will be required in the TTP/VASP protocol exchange.

The two processes are completely independent; communication between them is limited to the TTPA retrieving certificates which have previously been placed in a shared cache by the CSA.

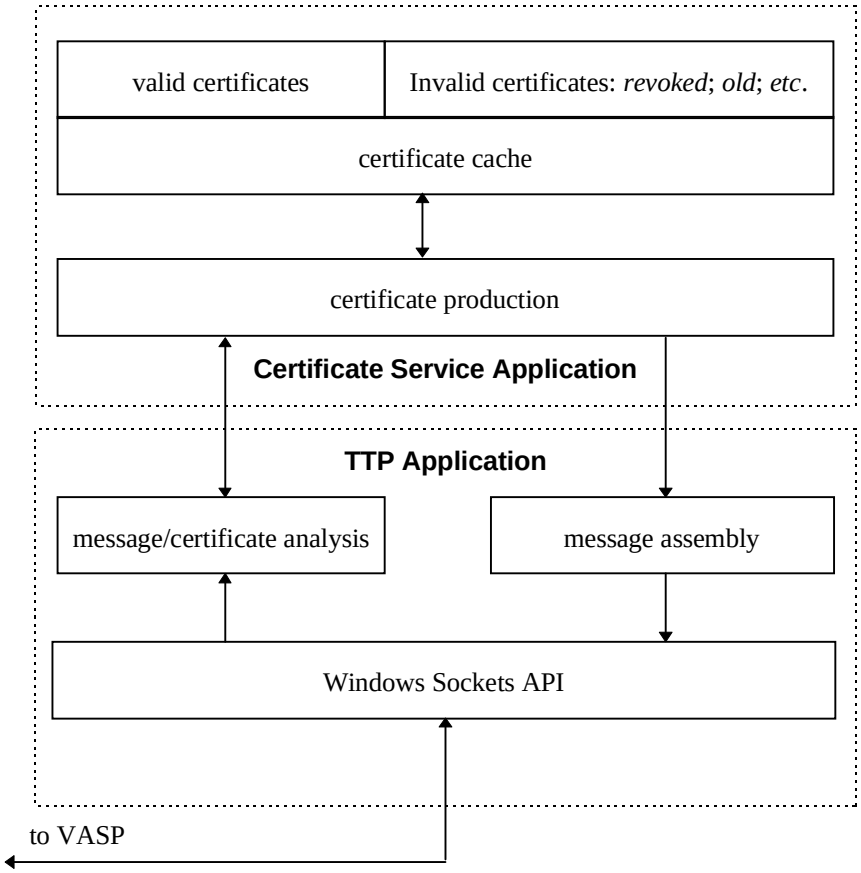

Figure 2.2.3 - Architecture of TTP software

2.2.1.5 Common Software Modules

2.2.1.5.1 Tracer

When processes in the demonstrator (clients) want to display some character-orientated information for the observer, they send this information to a tracer (server) process. In order to do this, clients must load the dynamic link library (DLL) **`tracedll.dll`**.

Before using the tracer, clients must register with the server process by calling the function **`int tracer_init(char *str)`**. The parameter **`str`** is an identifying string for the tracer's client and is ignored at this stage. If the function succeeds it returns **`0`**, in case of an error **`-1`** is returned. If the tracer process is not running, it is created.

The string **`str`** is sent to the tracer by calling the function **`int tracer_show(char *str)`**. If the function succeeds it returns **`0`**, in case of an error **`-1`** is returned.

The connection to the tracer is closed by calling the function **`int tracer_quit(void)`**. If the function succeeds it returns **`0`**, in case of an error **`-1`** is returned.

The communication between the tracer and the client processes uses Windows mailslots (similar to named pipes, however named pipes are not used because they are not supported under Windows 95). Windows mutual exclusions are used for the synchronization between the tracer and the client processes.

2.2.1.5.2 Cryptographic Library

The cryptographic library used by the joint demonstrator is ACRYL (Advanced CRYptographic Library) which is provided by Siemens AG in the form of several DLLs, the main DLL **`acryl.dll`** and auxiliary DLLs.

2.2.1.6 TTP and Secure Billing protocols

2.2.1.6.1 Introduction

The charging consists of two phases:

In the *initialisation phase*, the user and the VASP authenticate one another, and the user commits himself to a starting value for the micropayment scheme and a certain tariff by performing a digital signature on corresponding data.

In the *data transfer phase*, the user pays by releasing the pre-images of the starting value, so-called "ticks", which represent unit charges. The value of one unit charge is agreed upon in the initialisation phase. The "ticks" serve as proof to the VASP that the user incurred certain charges, because only the user could have generated them. They will be presented off-line by the VASP to the user's UMTS Service Provider (via the Network Operator) to settle the charges.

There are two variants of the payment process: *on-line*, meaning that a TTP is involved on-line in the payment process, in addition to the user and the VASP; *off-line*, meaning that the user and the VASP are the only entities involved on-line in the payment process.

There are three versions of the authentication protocol which is run in the initialization phase: versions A, B and C. Versions A and B are very similar. They realise the off-line version of the billing protocol as implemented in the first demonstrator. Authentication protocol version C realises the on-line variant of the payment process.

Protocols A and B were already described in [D10]. However, they have since undergone a modification and are thus also included in this deliverable.

2.2.1.6.2 Protocol goals

The goals to be achieved at the end of a successful run of protocols A, B and C are:

- mutual explicit authentication of *U* and *V*;
- agreement between *U* and *V* on a secret session key *K* with mutual implicit key authentication;
- mutual confirmation of key *K* between *U* and *V*;
- mutual assurance of key freshness (mutual key control);
- non-repudiation by *U* of data sent by *U* to *V* in the third message of the protocol;

- confidentiality of data sent by *U* to *V* in the third message of the protocol.

2.2.1.6.3 Prerequisites for Protocols and Choice of Cryptographic Parameters

For the **authentication and initialisation of payment protocol**, we have the following prerequisites and choice of cryptographic parameters:

- There is an elliptic curve cryptosystem *E* over *GF(p)* whose parameters *p* (prime defining the field), *q* (size of a cyclic subgroup of the curve), g_x and g_y (co-ordinates of a generator *g* of the curve), *a* and *b* (coefficients of the defining equation) are configurable. As default values, the parameter values in ISO/IEC DIS 14888-3, Annex C.2 are taken, where the cardinality *q* (of the cyclic subgroup of the elliptic curve) is in the order of 2^{129}.
- There is a function *f* mapping *E* onto the numbers in the range *[0..q-1]*. It is $f(Z) = Z_x \bmod q$.
- *V* has long-term secret and public key agreement keys *v* and g^v respectively, where *v* is a number in the range *[0..q-1]*, and $g \in E$ is as above.
- *U* possesses an asymmetric signature system with secret signature transformation Sig_U , secret key KU^- and public key KU^+. It is an AMV signature system based on the above elliptic curve *E*, as described in ISO/IEC DIS 14888-3. $Sig_U(M)$ denotes only the appendix.
- *U* and *V* possess a symmetric encryption function, where $\{ M \}_K$ is the encryption of message *M* with key *K*. We assume that the encryption algorithm is resistant against known cryptanalytic attacks such as code book attacks and chosen plaintext attacks. This function is DES-CBC.
- *U* and *V* possess a (pseudo-)random number generator. This RNG uses DES-OFB.
- *U* and *V* possess functions *h1*, *h2* and *h3* with the following properties:

  The functions *h1*, *h2*, and *h3* are compression functions (i.e. they map inputs of arbitrary finite lengths to fixed length outputs) which are easy to compute and satisfy:

  1. *h1* is a partial-preimage resistant, weakly computation resistant and weakly pseudo-random function.
  2. *h2* is a partial-preimage resistant, weakly computation resistant function.
  3. *h3* is collision resistant.

  (A function *h* is *weakly computation resistant (weak MAC-property)* if it is computationally infeasible to find a pair *(x, h(K, x))* without knowing *K*, provided that no other pair *(x', h(K, x'))* is known. A function *h* is *weakly pseudo-random* if, for secret random key *K* not used before and for known random *x* the output *h(K, x)* is indistinguishable from a random output.)

  The choices are as follows:
  h1 = h3 = RIPEMD-128, *h2(x)* = trunc*(40, h3(x))* where trunc*(n,y)* returns the *n* least significant bits of *y*.
- The identity *idV* of *V* is assumed to be known to *U* at the start of the protocol.
- There are parameters *ch_data*, *TV*, α_T and *IV* with the following significance: *ch_data* is data sent from *V* to *U* describing the applicable tariff, *TV* is a time-stamp generated by *V* (the UTC time of *V*), α_T and *IV* are random values generated by *U* for use in the payment protocol.

For **authentication and initialisation of payment protocol variant A** only:

- An authentic copy of the public key KU^+ of the asymmetric signature system of *U* is available at *V*. (It may have been distributed to *V* in an earlier run of protocol variant B.)

- An authentic copy of the public key agreement key g^V of *V* is available at *U*. (It may have been distributed to *U* in an earlier run of protocol variant B.)

For **authentication and initialisation of payment protocol variant B** only:

- There is a valid certificate *certU*, issued by a certification authority CA_U, on the public key KU^+ of the asymmetric signature system of *U*, available at *U*.
- There is a valid certificate *certV*, issued by a certification authority CA_V, on the public key agreement key g^V of *V*, available at *V*.
- *U* possesses the public key necessary to verify certificates issued by CA_V.
- *V* possesses the public key necessary to verify certificates issued by CA_U.

For **authentication and initialisation of payment protocol variant C** only:

- The TTP possesses an asymmetric signature system with secret signature transformation Sig_T, secret key KT^- and public key KT^+. If this is an asymmetric signature system with appendix then $Sig_T(M)$ denotes only the appendix.
- The TTP possesses the function *h3*, as defined above.
- *CertChain (A, B)* is a certificate chain on the public key of *B* which can be verified by an entity in possession of the public key of CA_A. (*A, B* may take the values *U, V; B* may also take the value *T*). If CA_A and CA_B coincide then *CertChain (A, B) = CertB.*
- *TT* is a time-stamp issued by the TTP.
- *cidA* is a unique identifier of a certificate on a public key of user *A*, e.g. a hash of the certificate or a combination of the identity of the issuer and the serial number.
- It is assumed that the user's identity is sufficient for the TTP to retrieve the appropriate certificate and that the user is in possession of the public key KT^+.

For the **payment protocols**, we have the following additional prerequisites and choice of cryptographic parameters:

- There is a public system parameter *T* which gives the maximum number of ticks to which the user can commit himself by one signature. *T* is equal to at most 2^{16}. For performance reasons, *T* may be set to a lower value. The default value for *T* is 2^{10}.
- There is a public family *F* of length-preserving one-way functions $F_{IV}: \{0,1\}^n \rightarrow \{0,1\}^n$, where *n* is a public system parameter and *IV* is an initialisation vector. (To be more precise, the functions F_{IV} need to be one-way on *T* iterates [Ped95])

The choices for the first and second demonstrators are $n = 64$ and $F_{IV}(x) = \text{trunc}(64, h(IV||x))$, where *h* is RIPEMD-128.

2.2.1.6.4 Authentication and Initialisation of Payment Protocol

The protocols are executed between a user *U* and a VASP *V*, and between the VASP *V* and a TTP *T*. Three variants of the protocols are available depending on the starting conditions:

- **Protocol Variant A:** *U* and *V* have authentic copies of one another's public keys.

- **Protocol Variant B:** *U* and *V* do not have authentic copies of one another's public keys but do have valid certificates on their own public keys.
- **Protocol Variant C:** *U* and *V* do not have authentic copies of one another's public keys or valid certificates on their own public keys. The on-line TTP provides the certificates to the user and the VASP.

Each variant is described in turn. For simplicity, only the information flow in the error-free cases is shown.

2.2.1.6.4.1 Protocol Variant A

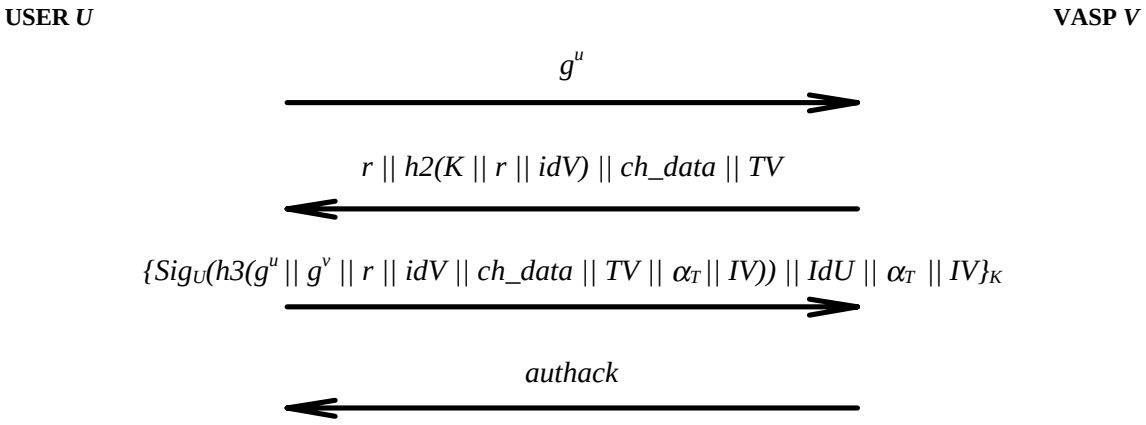

Figure 2.2.4 - Protocol variant A

operations at *U*:

1. generate random number *u*
2. compute g^u
3. send message $U \rightarrow V$: authreq: g^u

operations at *V*:

4. generate random number *r*
5. compute $K := h1(f((g^u)^v)||r)$
6. compute $h2(K \,||\, r \,||\, idV)$
7. send message $V \rightarrow U$: authcont: $r \,||\, h2(K \,||\, r \,||\, idV) \,||\, ch\_data \,||\, TV$

operations at *U*:

8. compute $K := h1(f((g^v)^u)||r)$
9. compute $h2(K \,||\, r \,||\, idV)$
10. display *ch_data* on the screen in a separate window; display warning on the screen if the difference *TV-TU* (*TU* = UTC time of *U*) is greater than a pre-defined *delta_t*; authcont_check = OK if received $h2(K \,||\, r \,||\, idV)$ equals $h2(K \,||\, r \,||\, idV)$ computed in step 9 and if *ch_data* is confirmed by human user via GUI (can be switched off).
11. compute $\{ IdU \}_K$, generate random α_0 and random *IV* and compute $\alpha_T = F_{IV}{}^T(\alpha_0)$. */*IdU = 64 bit string */*
12. compute $\{Sig_U(h3(g^u \,||\, g^v \,||\, r \,||\, idV \,||\, ch\_data \,||\, TV \,||\, \alpha_T \,||\, IV)) \,||\, IdU \}_K$

13. send message $U \rightarrow V$: authresp: $\{Sig_U(h3(g^u \,||\, g^v \,||\, r \,||\, idV \,||\, ch\_data \,||\, TV \,||\, \alpha_T \,||\, IV)) \,||\, IdU \,||\, \alpha_T \,||\, IV\}_K$

operations at *V*:

14. decrypt $\{Sig_U(h3(g^u \,||\, g^v \,||\, r \,||\, idV \,||\, ch\_data \,||\, TV \,||\, \alpha_T \,||\, IV)) \,||\, IdU\}_K$
15. retrieve KU^-. If $Sig_U(h3(g^u \,||\, g^v \,||\, r \,||\, idV \,||\, ch\_data \,||\, TV \,||\, \alpha_T \,||\, IV))$ verifies, set authresp_check = OK
16. store $Sig_U(h3(g^u \,||\, g^v \,||\, r \,||\, idV \,||\, ch\_data \,||\, TV \,||\, \alpha_T \,||\, IV))$, idU, $g^u \,||\, g^v \,||\, r \,||\, idV \,||ch\_data \,||TV \,||\alpha_T \,||\, IV$
17. set $j \leftarrow T$, $tck\_cnt \leftarrow 0$, $\alpha \leftarrow \alpha_T$
    / Initialisation of parameters of tick payment protocol. */*
18. send message $V \rightarrow U$: authack: $\{\ \}$
    / message contains no user data */*

operations at *U*:

19. set $j \leftarrow T$, $tick\_total\_U \leftarrow 0$
    / Initialisation of parameters of tick payment protocol. */*

2.2.1.6.4.2 Protocol Variant B

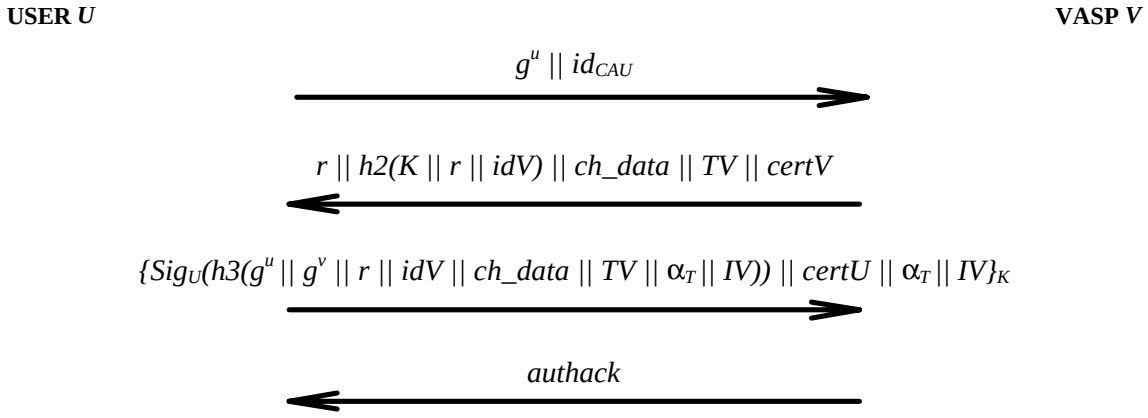

Figure 2.2.5 - Protocol variant B

The operations at *U* and at *V* are not described here as they are straightforward to analyse from the descriptions given for protocol A.

2.2.1.6.4.3 Protocol Variant C

The on-line TTP provides the following services in the course of the protocol:

- certificate distribution;
- provision of certificate chains using cross-certificates;
- time-stamping;
- assurance that a certain certificate was not revoked before a certain time;

- authorisation.

'Authorisation' means an assurance to the VASP that the user is authorised by his UMTS Service Provider to use the VASP's services, such that the risk of the user not paying for the service lies with the user's UMTS Service Provider and not with the VASP.

Authorisation may be implied by assurance of non-revocation. The idea is that a user's certificate will be revoked when he is no longer authorised to use services. Therefore, the user's UMTS Service Provider can be held liable by the VASP for costs incurred by the user if the VASP can prove that the user's certificate was not revoked and if the user's signature on the charge data can be verified.

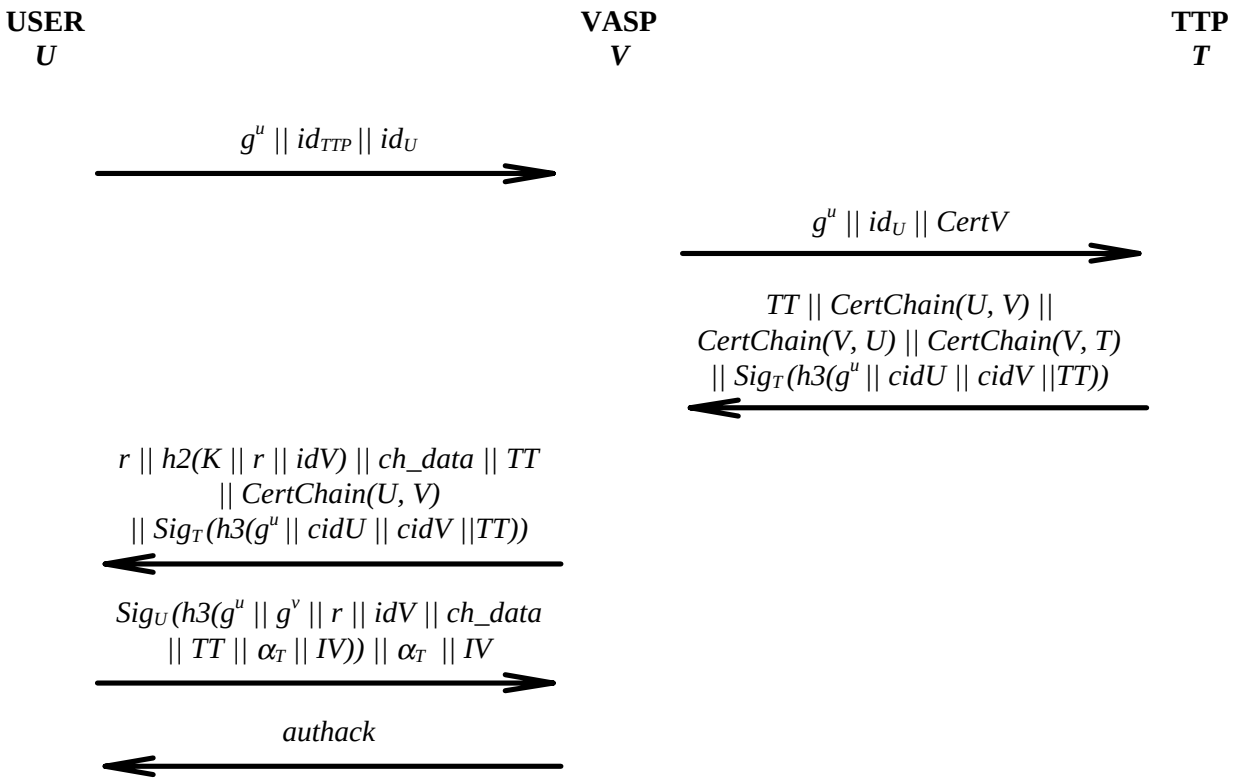

Figure 2.2.6 - Protocol variant C

operations at *U*:

1. generate random number u
2. compute g^u
3. send message $U \rightarrow V$: authreq: $g^u \,||\, id_{TTP} \,||\, id_U$
4. store u, g^u for later use in the protocol.

operations at *V*:

5. send message $V \rightarrow T$: ttpreq: $g^u \,||\, id_U \,||\, CertV$ to TTP with identity id_{TTP} sent in the first message.
6. store $cidV$, g^u

operations at *T*:

7. generate time-stamp *TT*
8. generate certificate chains *CertChain(U, V)*, *CertChain(V, U)* and *CertChain(V, T)*
9. generate signature $Sig_T(h3(g^u \,||\, cidU \,||\, cidV \,||TT))$
10. send message $T \rightarrow V$: ttpresp:
    $TT \,||CertChain(U, V) \,||\, CertChain(V, U) \,||\, CertChain(V, T) \,||\, Sig_T(h3(g^u \,||\, cidU \,||\, cidV \,||TT))$

operations at *V*:

11. verify *CertChain(V, U)*
12. retrieve *CertU* from *CertChain(V, U)* and *cidU* from *CertU*
13. retrieve public key of *T* from *CertChain(V, T)*
14. verify $Sig_T(h3(g^u \,||\, cidU \,||\, cidV \,||TT))$
15. generate random number *r*
16. compute $K := h1(f((g^u)^v)||r)$
17. compute $h2(K \,||\, r \,||\, idV)$
18. send message $V \rightarrow U$:
    authcont: $r \,||\, h2(K \,||\, r|| \, idV) \,||\, ch\_data \,||\, TT \,||CertChain(U, V) \,||\, Sig_T(h3(g^u \,||\, cidU \,||\, cidV \,||TT))$
19. store *TT, ch_data, K, CertU*

operations at *U*:

20. verify *CertChain(U, V)*
21. verify $Sig_T(h3(g^u \,||\, cidU \,||\, cidV \,||TT))$
22. compute $K := h1(f((g^v)^u)||r)$
23. compute $h2(K \,||\, r \,||\, idV)$
24. display *ch_data* on the screen in a separate window; display warning on the screen if the difference *TT-TU* (*TU* = UTC time of *U*) is greater than a pre-defined *delta_t*; authcont_check = OK if received $h2(K \,||\, r \,||\, idV)$ equals $h2(K \,||\, r \,||\, idV)$ computed in step 23 and if *ch_data* is confirmed by human user via GUI (can be switched off).
25. generate random α_0 and random *IV* and compute $\alpha_T = F_{IV}{}^T(\alpha_0)$
26. compute $Sig_U(h3(g^u \,||\, g^v \,||\, r \,||\, idV \,||\, ch\_data \,||\, TT \,||\, \alpha_T \,||\, IV)$
27. send message $U \rightarrow V$: authresp: $Sig_U(h3(g^u \,||\, g^v \,||\, r \,||\, idV \,||\, ch\_data \,||\, TT \,||\, \alpha_T \,||\, IV)) \,||\, \alpha_T \,||\, IV$
28. store *ch_data, K,* α_T*, IV*

operations at *V*:

29. retrieve KU^-. If $Sig_U(h3(g^u \,||\, g^v \,||\, r \,||\, idV \,||\, ch\_data \,||\, TV \,||\, \alpha_T \,||\, IV))$ verifies, set authresp_check = OK
30. store $Sig_U(h3(g^u \,||\, g^v \,||\, r \,||\, idV \,||\, ch\_data \,||\, TT \,||\, \alpha_T \,||\, IV))$, $idU, g^u \,||\, g^v \,||\, r \,||idV \,||ch\_data \,||TT \,||\, \alpha_T \,||$ *IV*

31. set $j \leftarrow T$, $tck\_cnt \leftarrow 0$, $\alpha \leftarrow \alpha_T$
    / Initialisation of parameters of tick payment protocol. */*
32. send message $V \rightarrow U$: authack: { }
    / message contains no user data */*

operations at *U*:

33. set $j \leftarrow T$, $tick\_total\_U \leftarrow 0$
    / Initialisation of parameters of tick payment protocol. */*

Remarks on **protocol version C**:

1. The identity of the user is not encrypted in the first message as it is assumed that the UMTS air interface is protected by underlying mechanisms. If it is of concern that an entity may impersonate a certain VASP to learn the user's identity then this could be addressed by a variant of the presented protocol.
2. The purpose of the signature sent in the third message differs for user and VASP: The VASP gets assurance that the user's certificate was not revoked at the time given by the time-stamp. He does not interpret the unique identifier of his own certificate nor the parameter g^u. It is assumed that the VASP has a reliable clock with which he can compare the time-stamp. The user, on the other hand, cannot generally be assumed to have access to a reliable clock (the protocol may be executed on a smart card). By including g^u in the signature, the user gets assurance that the time-stamp was created during the current protocol run and that the VASP's certificate was not revoked before the start of the current protocol run.
3. It is assumed that the user and the VASP know the unique identifiers of the certificates on their own public keys, so that they are able to verify the signature. If this may not be assumed then this unique identifier number has to be included in the third and the fourth messages in the clear.

2.2.1.6.5 Re-initialisation of Payment Protocol

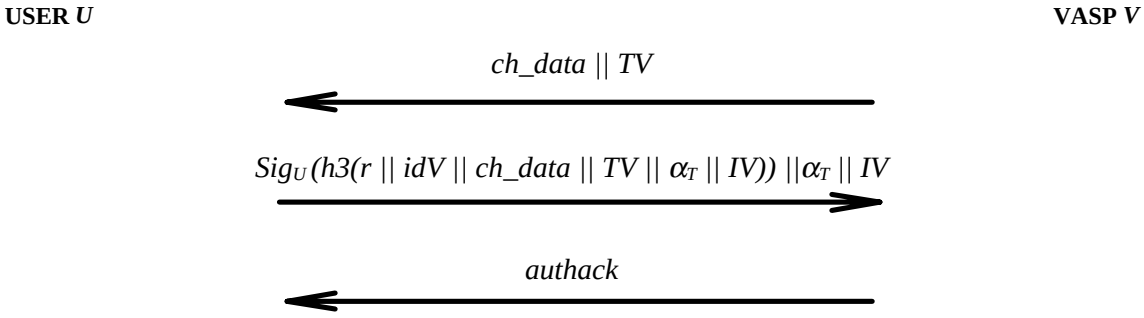

Figure 2.2.7 - Re-initialisation protocol

The detailed description of the protocol is straightforward from the description of protocol A, except perhaps for the following:

operations at *V*:

1. send message $V \rightarrow U$: reinitcont: $ch\_data \,||\, TV$
   / ch_data contains information on the tariff to be applied. TV is the UTC time of V. */*

operations at *U*:

2. display *ch_data* on the screen in separate window if *ch_data* is different from previously signed *ch_data*; display warning on the screen if the difference *TV-TU* (*TU* = UTC time of *U*) is greater than a pre-defined *delta_t*; authcont_check = OK if *ch_data* is equal to previously signed *ch_data* or *ch_data* is confirmed by human user via GUI, (the latter can be switched off).
3. generate random α_0 and compute $\alpha_T = F_{IV}{}^T(\alpha_0)$
4. compute $Sig_U(h3(r \,||\, idV \,||\, ch\_data \,||\, TV \,||\, \alpha_T \,||\, IV))$
5. send message $U \rightarrow V$: reinitresp: $Sig_U(h3(r \,||\, idV \,||\, ch\_data \,||\, TV \,||\, \alpha_T \,||\, IV)) \,||\alpha_T \,||\, IV$

operations at *V*:

6. retrieve KU−. If $Sig_U(h3(r \,||\, idV \,||\, ch\_data \,||\, TV \,||\, \alpha_T \,||\, IV)$ verifies, set reinitresp_check = OK
7. store $Sig_U(h3(r \,||\, idV \,||\, ch\_data \,||\, TV \,||\, \alpha_T \,||\, IV))$, $idU, r \,||\, idV \,||\, ch\_data \,||\, TV \,||\, \alpha_T \,||\, IV$
   / Do not overwrite previously stored transcripts */*
8. set $j \leftarrow T$, $tck\_cnt \leftarrow 0$, $\alpha \leftarrow \alpha_T$
   / Re-initialisation of parameters of tick payment protocol at V. */*
9. send message $V \rightarrow U$: authack: *{ }*
   / message contains no user data */*

operations at *U*:

10. set $j \leftarrow T$, $tick\_total\_U \leftarrow 0$
    / Re-initialisation of parameters of tick payment protocol at U. */*

2.2.1.6.6 Charge Ticks Protocol

USER *U* **VASP *V***

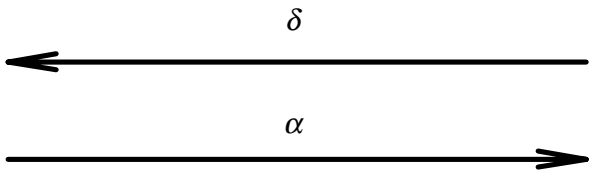

Figure 2.2.8 - Charge ticks protocol

Recall that initialisation of parameters is done at the end of the authentication and the re-initialisation protocols respectively.

notation at *U*:

/ T - j is the number of ticks already sent by U; tick_total_U is the number of ticks which has to be paid by U - according to the count of U; δ is the number of ticks whose payment is requested by V in the current run of the tick payment protocol. */*

notation at *V*:

/ T - j is the number of ticks requested by V; tck_cnt is the number of ticks correctly received by ;. α is the last tick received by V. */*

operations at *V*:

If $j \geq \delta$ then the charge ticks protocol is run, else the re-initialisation of payment protocol is run.

1. Set $\alpha' \leftarrow \alpha$
2. Set $j \leftarrow j - \delta$
3. Send message $V \rightarrow U$: ticks: δ

operations at *U*:

4. If $(j \geq \delta)$ and $(tick\_total\_U + j - T \geq \delta)$ then chtickreq_check = OK
   / On the user side, $j < \delta$ should not occur in a correct protocol run because the VASP also checks $j \geq \delta$. */*
5. Set $j \leftarrow j - \delta$
6. Set $\alpha \leftarrow F^j(\alpha_0)$
7. Send message $U \rightarrow V$: chtickresp: α

operations at *V*:

8. If $(\alpha' = F^\delta(\alpha))$ then chtickresp_check = OK
9. Set $tck\_cnt \leftarrow tck\_cnt + \delta$
10. Store α, *tck_cnt*
    / α, tck_cnt should be continuously stored, not only at the end of the session, otherwise the paid ticks are lost when the program crashes. Previous values are overwritten. */*

2.2.1.7 Certificate data structures

2.2.1.7.1 Introduction

In the context of asymmetric cryptography, two entities wishing to communicate securely with one another may need to:

- verify the identity of one another,
- verify the signature of one another,
- encrypt a message by using the public encipherment transformation of one another, or
- agree on a session key by using the public key agreement key of one another.

For these purposes, each entity must obtain the other's public key from a source that it trusts. Such a source, called a Certification Authority (CA), uses a signature algorithm to certify the public key, producing a certificate. This CA needs to have its own digital signature key pair.

A certificate must have the following three properties:

- only an authorised CA can issue a valid certificate;
- any entity with access to the public signature verification key of the CA can recover the public key which was certified;
- no entity other than the CA can modify the certificate without this being detected (certificates are unforgeable).

In the ASPeCT model of future mobile telecommunications systems, the certification of public keys is provided by TTPs identified in ASPeCT Deliverable 2 [D02], and used to support security related services, e.g. end-to-end security services and secure billing services. The following five types of entity may need to have public keys certified:

- mobile users,
- Network Operators (NOs),
- Service Providers (SPs),
- VASPs,
- TTPs.

ASPeCT TTPs [D07] will be responsible for the following tasks to support certification of public keys for individual entities:

- generation of a certificate,
- maintenance of a Directory Information Base (DIB), which is used to store certificates,
- management of a Directory Information Tree (DIT), which is used to issue a particular certificate to a particular entity via a suitable certification path,
- revocation of an invalid certificate, and
- generation and maintenance of Certificate Revocation Lists (CRLs).

Here, we are concerned with the generation of certificates, including client certificates, TTP certificates and cross certificates as described below:

- clientCertificate: a certificate for the public key of a client (e.g., mobile user, VASP, NO and SP),
- TTPCertificate: a certificate for the public key of a TTP (in particular, this TTP is a CA),
- crossCertificate: a certificate for another certificate of a public key, which includes both forward certificate and reverse certificate.

For simplicity, we make use of the same data structure for client and TTP private keys as well as for certificates on client and TTP public keys. The main difference with a private key certificate is that it is not necessary to sign a secret key. Instead of the signature, other origin authentication, data integrity and confidentiality techniques will be used.

Each certificate for a public key consists of two parts: certificate type identifier and signed certificate information sequence.

2.2.1.7.2 Types of Certificate

We have two types of certificates for ASPeCT, depending on which signature mechanism is used. The leftmost byte of the certificate string is the certificate type identifier.

The first type of certificate makes use of an RSA-signature based on ISO/IEC 9796-2 (Draft International Standard) [ISO96b]. Its certificate type identifier is 00 (hex). Its signed certificate information sequence includes a signed recoverable string and a non-recoverable part.

The second type makes use of an AMV-signature based on ISO/IEC 2nd CD 14888-3 [ISO96a]. Its certificate type identifier is 01 (hex). Its signed certificate information sequence is the certificate information sequence itself together with an appendix, which is the signature of the sequence.

Within ASPeCT, these two types of certificates make use of one single certificate information sequence format as described below.

2.2.1.7.3 Certificate Information Sequence Format

In the description of the certificate information sequence, *issuer* denotes the entity issuing the certificate authoritatively; and *subject* denotes the entity holding the private key, for which the corresponding public key is being certified.

The certificate information sequence includes the basic certificate information and a set of extended attributes providing other optional information about both the subject and the issuer. Table 2.2.1 shows the certificate information sequence format for ASPeCT.

Field	Contents	Description	Length
1.	Map field	This field gives the map which fields and options will be presented in the certificate. It includes the following 8-bit information: 1 public key (1) / secret key (0) 2 issuer public key identifier present (1) / not present (0) 3 issuer identifier format: hash (1) / plain (0) 4 subject private key usage period present (1) / not present (0) 5 subject identifier format: hash (1) / plain (0) 6 subject key usage present (1) / not present (0) 7 cross certificate attribute present (1) / not present (0) 8 certification path present (1) / not present (0)	1 byte binary
2.	Version	The version number of the certificate. The version that we will start with is V1.0. This field will therefore contain the value 10 (hex).	1 byte binary
3.	Serial Number	Unique number of the certificate, assigned by the issuer.	12 bytes binary
4.	Public key identifier	Optional. Unique identifier of the public key (e.g., as key updating occurs) to be used to verify the signature on this certificate.	1 byte binary
5.	Issuer Identifier	There are two options for the issuer identifier: 1 the binary issuer identifier, like a serial number 2 the plain issuer identifier.	 16 bytes binary 15 + 30 bytes binary
6.	Validity	Including four dates: 1 the date before the certificate is not valid, 2 the date after the certificate is no longer valid. 3 optional private key usage period: including the date before and the date after as well. It is used when the subject private key usage period is not the same as the public key validity.	 6 bytes binary 6 bytes binary 6 + 6 bytes binary
7.	Subject Identifier	There are two options for the subject identifier: 1 the binary subject identifier, like a serial number 2 the plain subject identifier.	 16 bytes binary 15 + 30 bytes binary
8.	Subject key usage	The usage of the subject key being certified includes: 0 = digital signature, 1 = data encryption, 2 = key agreement, 3 = key certificate signature, 4 = CRL signature.	1 byte binary
9.	Cross	Optional. Two situations:	

	certificate attributes	1 The public key certified will be used to sign another certificate (issuer identity and certificate serial number). 2 The public key used to signed this certificate was certified by another certificate (issuer identity and certificate serial number).	30 bytes binary 30 bytes binary
10.	Certificate path attributes	Optional. Two subfields: 1 Path length - the number of related certificates. 2 A list of subject identifiers included in the certificate path.	 1 byte binary 16 byte binary per each subject name
11.	Subject public key information	An algorithm type identifier plus a public key value for the subject. Subfield 1: algorithm type identifier……..…. 0 = RSA 1 = elliptic curve 2 = Diffie-Hellman other unspecified If algorithm type identifier = 0 Subfield 2: modulus length of key in bits……..….. Subfield 3: exponent length of key in bits……..…. Subfield 4: key value: first modulus, then exponent of key……..….. If algorithm type identifier = 1 Subfield 2: length of x-coordinate of key in bits…… Subfield 3: key value: first x-coordinate, then y-coordinate of key…….. Subfield 4: parameter set identifier……..... If algorithm type identifier = 2 Subfield 2: length of key value in bits…….... Subfield 3: key value: g^x mod p……..... Subfield 4: parameter set identifier……..	 1 byte binary 2 bytes binary 2 bytes binary (sum of (values of subfields 2 and 3)+7)/8 bytes binary 2 bytes binary 2 * ((value of subfield 2+7)/8) bytes binary 1 byte binary 2 bytes binary (value of subfield 2+7)/8 bytes binary 1 byte binary

Table 2.2.1 - A certificate information sequence format

2.2.1.7.4 Signature Mechanism

Within ASPeCT we have two models of signature used to sign a certificate information sequence. The related certificate type identifiers are assigned as 00 and 01, respectively.

2.2.1.7.4.1 RSA-signature based on ISO/IEC 9796-2 (Draft International Standard)

Assume that there exist a public modulus n of k bits ($512 \leq k \leq 1024$), a private signature exponent s and a corresponding public verification exponent v. These values may be different for each certificate, and must have the usual properties required for operation of the RSA algorithm, namely that n is a product of two large prime numbers p and q, that v and $(p-1)(q-1)$ are relatively prime, and that sv is congruent to 1 modulo $lcm((p-1)(q-1))$. Note that the two primes, p and q, should be discarded, and never revealed after key production.

Suppose that a certificate information sequence M, as described in the above section, is of m bits. There are two possible cases:

1. the entire message can be recoverable from the signature;
2. the message shall be split into two parts: a non-recoverable part M_x of x bytes, where x is a positive integer, and a recoverable part M_r of m-$8x$ bits.

We make use of RIPEMD-160 hash function to compute a hash-code H of 160 bits from the entire message M. The recoverable string S_r of n bits is then constructed as shown in Table 2.2.2.

Left adaptation two bits	More-data bit one bit	Padding field one or more bits	Data field m bits or m-8x bits	Hash-code field 160 bits	Trailer 1 or 2 bytes
01	0 if M_x empty 1 otherwise	0…01	M if M_x empty M_r	H	'BC' or 'XYCC'

Table 2.2.2 - A recoverable string S_r

If the hash-function in use is either implicitly known or coded inside the message, then the trailer shall consist of one byte set at 'BC'. If the rightmost byte is set at 'CC', then the trailer shall consist of two bytes where the leftmost byte is the hash-function identifier. For example, '31' denotes RIPEMD-160.

Note that some bits of this string have to be changed in the following way.

- The leftmost nibble shall remain unchanged.
- Every subsequent nibble equal to '0', if any, shall be replaced by a nibble set to 'b'.
- The first subsequent nibble not equal to '0' is the border nibble: it carries the border bit; it shall be exclusive-ored with 'b'.
- The remaining bits shall remain unchanged.

We then sign the recoverable string S_r , in both cases using the signature function under control of secret signature key. The signature is $Sign_s(S_r) = S_r^s \bmod n$.

The signed certificate information sequence shall be either

- the signature alone, if M_x is empty in the first case, or
- the non-recoverable part M_x together with the signature in the second case.

A certificate shall be as shown in Table 2.2.3.

	Certificate type identifier one byte	*Signature* *k* bits	*Non-recoverable data* *8x* bits
first case, $k+8$ bits	*0*	$Sign_s(S_r)$	
second case, $k+8x+8$ bits	*0*	$Sign_s(S_r)$	M_x

Table 2.2.3 - Certificate format based on RSA-signature

The verification process is a reversed procedure of the signature.

2.2.1.7.4.2 AMV-signature based on ISO/IEC 2nd CD 14888-3

This signature mechanism is an ElGamal-type signature scheme based on elliptic curves over finite fields.

In the description of the signature mechanism, we make use of the following notation.

ε a finite commutative group

$\#\varepsilon$ the cardinality of ε

p a divisor of $\#\varepsilon$

g an element of order p in ε

X secret signature key, $0 < X < p$

Y public verification key, $Y = g^X$

f a map from elliptic curve points to Z_p

h a hash function

Suppose that a certificate information sequence M is of m bits. The signature equation (the Agnew-Mullin-Vanstone equation) is $RK\text{-}SX\text{-}H = 0 \bmod p$, with:

K a random number

R the first part of the signature, $R=f(g^K)$

H a hash-code, $H=h(M)$

S the second part of the signature, $S=(RK-H)X^{-1} \bmod p$

Σ the signature, $\Sigma=(R,S)$

The length of the certificate depends on the length of p. Annex C.2 of ISO/IEC 2nd CD 14888-3 gives an example with parameter length 129 bits and hash value of length 128 bits. Assume that these numbers can be used in our certificate, the certificate shall be as shown in Table 2.2.4.

Type of certificate	Length of Certificate information sequence (in bits)	Certificate information sequence	Length of R and S (in bits)	Appendix
one byte	2 bytes	m bits	2 bytes	256 bits
1	L_M	M	L_R	R,S

Table 2.2.4 - Certificate format based on AMV-signature

The verification process is as follows.

$[g^K]' = Y^{(S/R) \bmod p}\ G^{(H/R) \bmod p}$

$R' = f([g^K]')$

If $R'=R$, the signature is verified successfully.

2.2.1.8 Communication Interfaces

Setting up the communication, as well as sending and receiving Protocol Data Units (PDUs), is accomplished using the Windows Sockets Application Programming Interface.

The messages or PDUs to be sent between the ASPeCT applications have the following format:

	id	**length**	**data**
size	1 byte	2 bytes	specified by **length**

The values for **id** (8 bits) are:

Authentication protocol:

0x01 for the first message from the user to the VASP (authreq)

0x02 for the message from the VASP to the TTP (ttpreq)

0x03 for the message from the TTP to the VASP (ttpresp)

0x04 for the first message from the VASP to the user (authcont)

0x05 for the second message from the user to the VASP (authresp)

0x06 for the second message from the VASP to the user (authack)

Re-initialisation protocol:

0x11 for the first message from the VASP to the user (reinitcont)

0x12 for the message from the user to the VASP (reinitresp)

0x13 for the second message from the VASP to the user (authack)

Charge protocol:

0x21 for the message from the VASP to the user (ticks)

0x22 for the message from the user to the VASP (chtickresp)

Error code:

0xFF

The **length** field (16 bits) contains the length of the complete message (including id and length) in bytes as an unsigned 16 bit integer.

The **data** field contains a sequence of BitStrings, where each data item in the message is one BitString.

Each BitString consists of a length field of four bytes, followed by the value. The meaning of each data item is implicit given its position within the PDU.

length	**value**
4 bytes	specified by **length**

The length field contains the length of the value field of the BitString in bits as an unsigned 32 bit integer.

So the messages of the authentication and initialisation of payment protocol variant A contain 1, 4, 4, 0 BitStrings.

All data is sent in the internal PC (Intel) format, so no conversion is needed before sending / after receiving.

2.2.2 Trial Methodology

2.2.2.1 Overview

The ASPeCT TTP and secure billing trial will be associated with the EXODUS Multimedia Trial. The purpose of the field trial is to show that the proposed security mechanisms work in an experimental UMTS environment.

The TTP and secure billing trial will be carried out in the form of a 'field trial' which will consist of a group of real users using the system with the assistance of ASPeCT and EXODUS staff. Users will report on their experiences on using the system by filling in a questionnaire.

The questionnaire will be used to evaluate the user acceptability of system performance and quality of service as perceived by users who interact with the software through the GUI. It is noted however that the focus of the trial is not on the ergonomic aspects of the user interface.

The field trial will also be used to analyze the technical feasibility of the system in absolute terms by gathering performance measurements from log files generated during the course of the trial.

ASPeCT is responsible for the technical side of the trial as far as the ASPeCT TTP and secure billing applications are concerned. EXODUS is responsible for the technical side of the trial as far as the UMTS platform and terminals are concerned.

EXODUS will be in charge of the user groups. Both ASPeCT and EXODUS will be involved in the evaluation of the trial.

The trial requirements are:

- ASPeCT will have two days in the first two weeks in March to test the trial software on the EXODUS trial platform operated by SwissCom.
- The trial will take place during April 1998 in Oensingen, Switzerland.
- There will be:
  10 expert users;
  25 other users with some general experience with PCs and the WorldWideWeb.
- Each user will have:
  - 1 hour in order to read the documentation;
  - 30 minutes to run the demonstration;
  - 30 minutes to answer the questions.

ASPeCT will provide:

- Documentation including information about the demonstrator and instructions for the user how to carry out the trial;
- Questionnaire for the users to complete;
- Methodology to evaluate the user acceptability based on results from the questionnaire.

ASPeCT(SwissCom) will, in collaboration with EXODUS:

- give ASPeCT the opportunity to test the demonstrator before March 14;
- carry out the trial (instruct users, organise trial logistics);
- take care of the users (thus ASPeCT presence is not necessary during the trial);
- evaluate the questionnaires and write a summary report.

2.2.2.2 Trial Configuration

2.2.2.2.1 Overview

The TTP and secure billing field trial involves three entities: a mobile user, a VASP and a TTP. The EXODUS site for the field trial is Oensingen. The mobile user, VASP and TTP have fixed broadband access to the EXODUS experimental UMTS platform.

The configuration for the trial is shown in Figure 2.2.9. The ASPeCT TTP and secure billing software will exist on the EXODUS terminal PCs. The mobile user will implement some of the TTP and secure billing application on an ASPeCT smart card. The VASP and the TTP will not implement any of the TTP and secure billing application on a smart card. An ASPeCT card reader will be attached to the EXODUS terminal PC representing the mobile user.

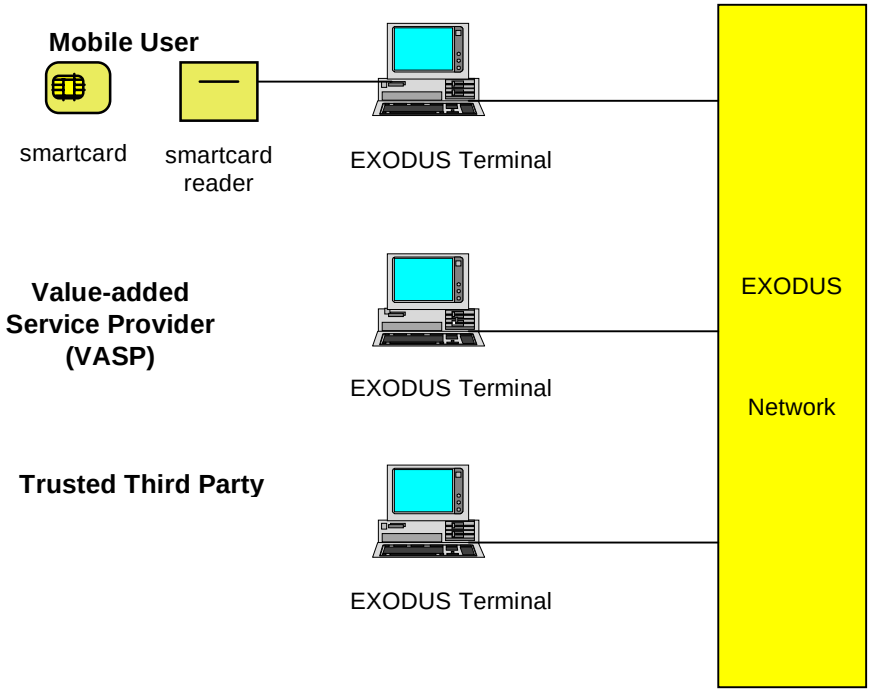

Figure 2.2.9 - Trial configuration

The trial will consist of three phases:

Phase 1: Data connections are set up between the user and the VASP, and between the VASP and the TTP.

Phase 2: The TTP and secure billing applications are launched.

Phase 3: The TTP and secure billing services applied to a value-added information service are used. This phase requires the user to insert the TTP and secure billing trial smart card.

2.2.2.2.2 Relationship Between TTP and Secure Billing Trial and Authentication Trial

The TTP and secure billing demonstrator and the authentication demonstrator are not to be trialled simultaneously; the ASPeCT user-to-network authentication will be switched off during the TTP and secure billing trial.

2.2.2.2.3 Integration With EXODUS Platform

The configuration for the field trial is shown in Figure 2.2.9. EXODUS provides three multimedia terminals connected to the EXODUS network, all three terminals are located in Oensingen.

The interface between the ASPeCT software and the EXODUS software is Windows Sockets 1.1. This means that the communication between the ASPeCT applications uses Windows Sockets 1.1 provided by EXODUS. The communication structure for the TTP and secure billing trial is shown in. Parts to be provided by EXODUS especially for this trial configuration are horizontally hatched, other parts to be provided by EXODUS are vertically hatched. The connection between the user and the VASP is semi-permanent, i.e. established by management procedures and not by a call set-up. The semi-permanent connection is realised by the driver for the EXODUS ATM adapter and the ATM switch between the user and the VASP. The EXODUS terminal representing the VASP will have to maintain two simultaneous connections, one with the user and one with the TTP. For the connection with the TTP, EXODUS provides InterWorking Units (IWUs), one on the VASP PC and one on the TTP PC, which realises Windows Sockets for the ASPeCT applications over the EXODUS network.

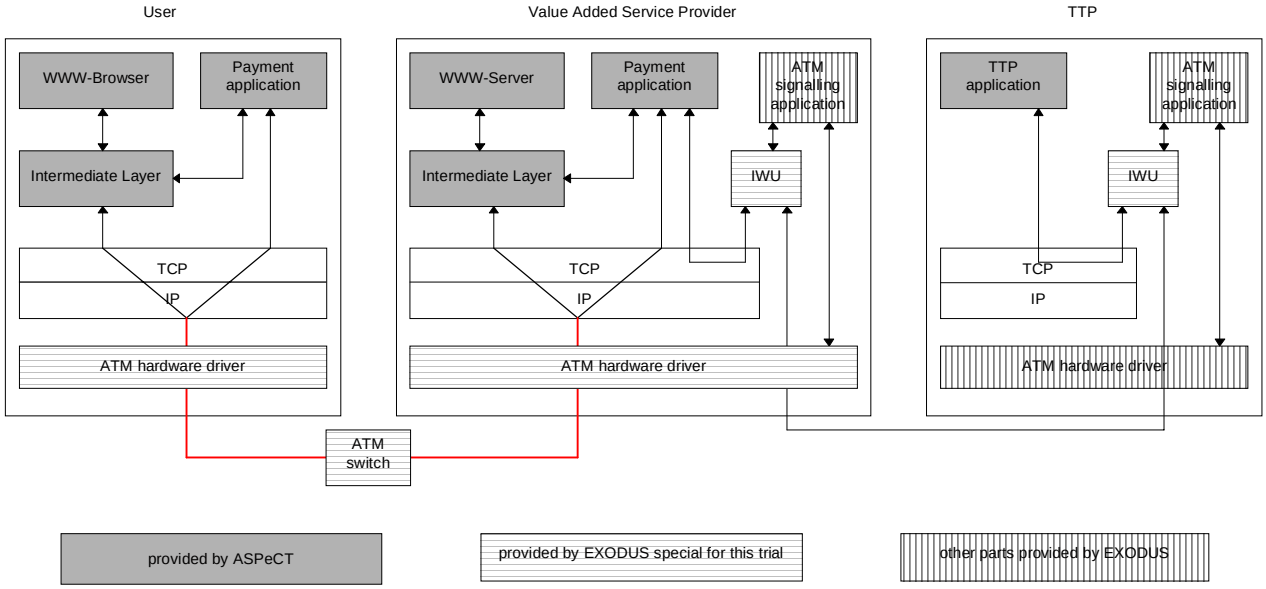

Figure 2.2.10 - Trial communication structure

ASPeCT will provide the ASPeCT TTP and secure billing applications, as marked in. Furthermore ASPeCT will provide the smart card reader for the user terminal as well as the smart cards for the trial users. These smart cards cannot be used for user authentication as carried out in the authentication trial. The EXODUS terminals for the user, the VASP and the TTP must provide network access without the smart card used in the authentication trial.

2.2.2.3 Trial Scenarios

On the VASP terminal a WWW server application will provide a value-added information service trial users. The trial users at the user terminal will use the WWW browser application to retrieve HTML documents which are of particular interest to them using the HTTP protocol.

As soon as the browser connects to the server, the ASPeCT payment application on the user terminal will connect to the ASPeCT payment application on the VASP terminal and they will execute the authentication

and initialisation of payment protocol, assuring each other's identity and establishing all the data needed for the charge protocol to be executed and for the on-line credit-based payment of the retrieved information. In the course of this protocol the user's smart card is required. In return for the requested information, the server payment application asks the user for a payment. The user's payment application responds to this request by making the payment.

In the window of the payment application on the user PC and on the VASP PC, all relevant information about the information service and the secure billing protocols associated with the current session are displayed. All this information is also stored in the user's log file so he can check how much money he spent in all his sessions.

The exercise of error conditions and error messages resulting from certificate problems (revoked, out of date etc.) is not included in the trial scenario for the users, but will be included in a separate demonstration.

2.2.2.4 User Registration

Before being able to use the value-added service, mobile users in the trial must register with a TTP. There are several possibilities for the user registration:

1. User identifications, secret / public key pairs with certificates and smart cards are prepared in advance for the maximum number of users for the trial. The disadvantage of this approach is the large number of smart cards of which each is used only for one demonstration.
2. User identifications and secret / public key pairs with certificates are prepared in advance for the maximum number of users. There is only a limited number of smart cards and the smart cards are personalized for every trial user before he uses the demonstrator. The disadvantage of this approach is that this is not a realistic demonstration of the user registration and certification by a TTP.
3. It would be a more realistic approach to have a user registration in such a way that the user sits in front of the TTP PC giving an identification (name, address, ...) and the TTP assigns a unique identification for the value-added service or the network to be used, generates a certificate, and personalizes a smart card for the user. The secret key could be generated by the TTP and stored on the smart card. The secret key could also be generated by the smart card together with the public key which would be sent to the TTP, in which case the secret key would never be known outside of the smart card.

2.2.2.5 User Documentation and Training

The TTP and secure billing field trial will involve a number of users retrieving and paying for value-added information from a VASP. In order to evaluate user acceptability, it is necessary to adopt a structured approach in order that meaningful results can be obtained from users in the trial. To achieve this, users will be given documentation outlining a procedure which should be followed during the course of the trial. The documentation will also provide the users with background information which they will need in order to fully appreciate the functionality provided by the demonstration software. After reading the background information and following the instructions the user should then be able to provide feedback by means of a questionnaire.

Two sets of documentation will be provided for each of the two user groups in the trial. The 'non expert users' will be given high-level background information on the scope and purpose of the trial. They will then be asked to complete a relatively straightforward questionnaire based on a list of actions they will be required to perform using the demonstration software. The 'expert users' on the other hand will be provided with more detailed information on the protocols and mechanisms used in the trial. They will then be asked to complete a more complex questionnaire based on a list of actions they will be required to perform using the demonstration software.

2.2.2.6 Trial Evaluation

The trial will be evaluated both in terms of the technical feasibility of the solution and in terms of the (perceived) user acceptability of the solution.

2.2.2.6.1 Evaluation of Technical Feasibility

In order to help evaluate the technical feasibility of the solution, performance measurements will be made during the TTP and secure billing field trial.

All measurements will be obtained from the analysis of data written to log files in each of the three entities in the trial; the user, the VASP and the TTP. It is necessary to be able to deduce the required performance indicators and parameters from the measurements contained in the three log files.

2.2.2.6.2 Evaluation of User Acceptability

In order to help evaluate the user acceptability of the solution, questionnaires will be distributed to users as part of the TTP and secure billing trial.

The questionnaires will be used to help evaluate the acceptability of system performance and quality of service as perceived by users.

Two sets of questionnaires (and accompanying documentation) will be produced for each of the two user groups in the trial. The 'non expert users' will be asked questions on ease of use, performance, security awareness. The 'expert users' will be presented with a more complete picture of how this solution would fit into a 'real' UMTS scenario. They will thus be asked more in depth questions relating to charging and billing for value-added services and other electronic commerce applications in UMTS. Furthermore, the 'expert' users will be asked more detailed questions about their trust and confidence in the underlying security protocols used to realise the secure billing application.

The user-related documentation, together with the questionnaire results will be included as part the final evaluation material for the trial.

2.2.3 Realisation

2.2.3.1 Graphical User Interface

2.2.3.1.1 User

On the client PC there are two ASPeCT applications with different graphical user interfaces:

- The secure billing application for viewing information on the current communication session (possibly including the display of detailed information in the character-orientated tracer, cf. Section 2.2.1.5.1);
- A separate application for off-line viewing of information on previous communication sessions.

The secure billing application displays in its main window information about the current communication session with a particular VASP. This information is stored in a file which can, for example, be used to check the correctness of bills that the VASP might send.

In addition, a separate application is available to interpret this file and display a listing showing all the VASPs the user has visited in the past, together with the accumulated payments for all the visits. The user then has the possibility of looking at a more detailed listing for a selected VASP showing the payment parameters for all sessions with that particular VASP. Furthermore, the user will have the possibility to delete old information which is no longer of any interest to him.

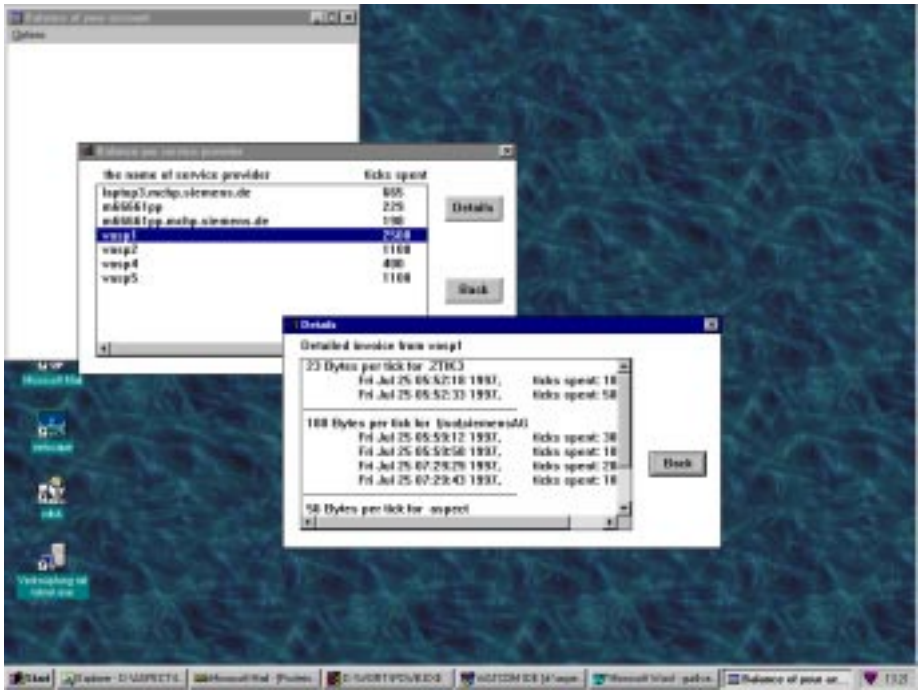

Figure 2.2.11 - GUI for User

2.2.3.1.2 VASP

On the VASP's PC there will be one ASPeCT application with a graphical user interface, namely the secure billing application. The GUI of the secure billing application on the VASP will display in its main window information about the current session with one particular user. This information is also stored in a file, such that detailed bills can be generated at a later date.

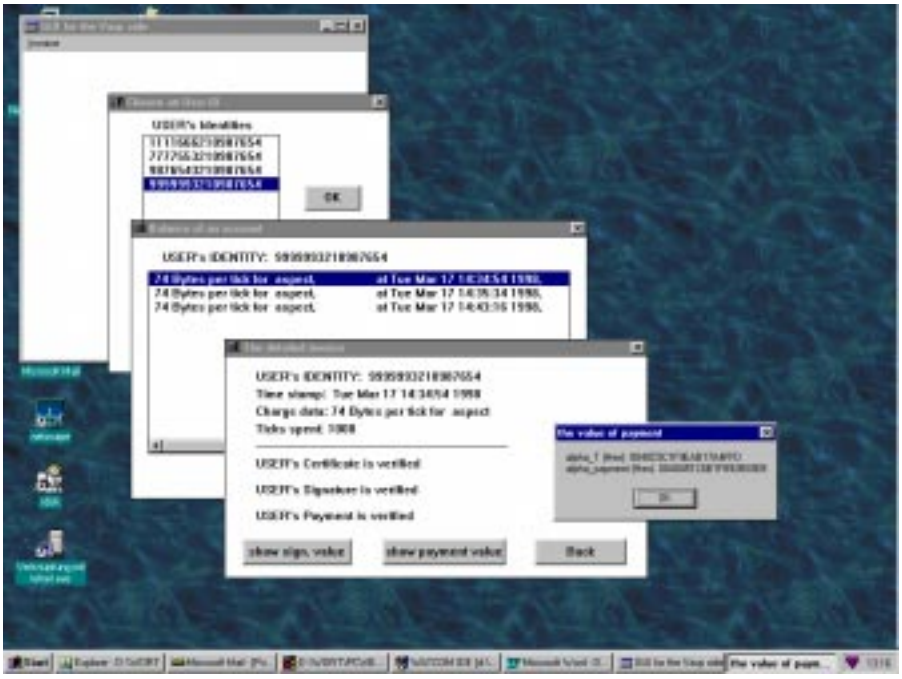

Figure 2.2.12 - GUI for VASP

2.2.3.1.3 TTP

As described above, there are two applications which run on the TTP PC:

- the TTP application (TTPA), which runs during the interchanges with the VASP;
- a Certification Service application (CSA) which generates certificates off-line; the CSA can be selected as a virtual Certification Authority (CA) for T, V, or Users, and will sign with the appropriate key.

There is no operator interaction with the TTP during the demonstration; the user interface provides only monitoring or review of the progress of the protocols via the *tracer*.

The user interface of the CSA provides the following facilities:

	user action	input parameters	result	errors/warnings
(1)	select CA for client	*T, V,* U_n		<*CA* already exists, re-initialize?>
(2)	generate P_{CA}, (for CA selected in (1))		Public key for signature of *CertX*	< P_{CA} already exists, overwrite it?>
(3)	request certificate for X (in case of U_n, this is effectively *user registration*)	*idX*	*CertX* signed with P_{CA} (where *CA* is the one appropriate to *X*) is placed in *certificate cache*	< *CertX* already exists, overwrite it?>
(4)	make *CertX* using P_X supplied on diskette	*X*, (P_X is read from diskette)	*CertX* signed with P_{CA} (where *CA* is the one appropriate to *X*) is placed in *certificate cache*	<*bad input*: the key is not an appropriate one> <*Certificate cache* cannot be accessed. *CertX* cannot be written in the *cache*>
(5)	revoke certificate for X	*idX*	*bad* flag on *CertX* in *certificate cache*	< *CertX* does not exist in the *certificate cache*> <*Certificate cache* cannot be accessed. *CertX* cannot be revoked from the *cache*>
(6)	update user smart card for U_n	idU_n	private key for U_n written to current smart card	<*private key* already exists, overwrite it?>
(7)	output *CertX* to diskette			

Table 2.2.5 - CA operations

2.2.3.2 Certificates

2.2.3.2.1 Certificate Information Sequence Profile

The second demonstrator makes use of the AMV certificate, as described in Section 2.2.1.7.4.2.

In the second demonstrator, we use a profile of the ASPeCT certificate information sequence format described in Table 2.2.1. The profile ignores most of the options in the ASPeCT certificate information sequence format. The certificate information profile for the second demonstrator is described in Table 2.2.6.

Thus if this basic certificate and the AMV-signature is used, the total length of the certificate will be 154 bytes [3], if a cardinality (of the cyclic subgroup of the elliptic curve) of 129 bits is used.

Field	Contents	Description	Length
1.	Map field	For public key – 10111000 For secret key – 00111000	1 byte binary
2.	Version	10 (hex)	1 byte binary
3.	Serial Number	Unique number of the certificate, assigned by the issuer	12 bytes binary
4.	Public key identifier	No	
5.	Issuer Identifier	The binary issuer identifier	16 bytes binary
6.	Validity	Including two dates: 1 the date before which the certificate is not valid, 2 the date after which the certificate is no longer valid.	 6 bytes binary 6 bytes binary
7.	Subject Identifier	The binary subject identifier	16 bytes binary
8.	Subject key usage	The usage of the subject key being certified includes: 0 = digital signature, 1 = data encryption, 2 = key agreement, 3 = key certificate signature, 4 = CRL signature.	1 byte binary
9.	Cross certificate attributes	No	
10.	Certificate path attributes	No	
11.	Subject public key information	An algorithm type identifier plus a public key value for the subject. Subfield 1: algorithm type identifier ……. 0 = RSA 1 = elliptic curve 2 = Diffie-Hellman other unspecified If algorithm type identifier = 0 Subfield 2: modulus length of key in bits……..…… Subfield 3: exponent length of key in bits……..…… Subfield 4: key value: first modulus, then exponent of key …….. If algorithm type identifier = 1 Subfield 2: length of x-coordinate of key …….. Subfield 3: key value: first x-coordinate, then y-coordinate of key …….. Subfield 4: parameter set identifier…… If algorithm type identifier = 2 Subfield 2: length of key value in bits…… Subfield 3: key value: g^x mod p……. Subfield 4: parameter set identifier	 1 byte binary 2 bytes binary 2 bytes binary (sum of (values of subfields 2 and 3)+7)/8 bytes binary 2 bytes binary 2 * ((value of subfield 2+7)/8) bytes binary 1 byte binary 2 bytes binary (value of subfield 2+7)/8 bytes binary 1 byte binary

Table 2.2.6 - A certificate information sequence profile for the 2nd demonstrator

3 Other certificate lengths may be observed, however this figure represents the general case.

2.2.3.2.2 Certificate Chains for Second Demonstrator

Certificate chains for the second demonstrator are based on the definition in Section 2.2.1.6.3:

> *CertChain (A, B)* is a certificate chain on the public key of B which can be verified by an entity in possession of the public key of CA_A. (*A* may take the value *U* or *V*, whereas *B* may take one of the values *U*, *V* or *T*). If CA_A and CA_B coincide then *CertChain (A, B) = CertB.*

Two possibilities are provided in the general certificate definition Table 2.2.6:

- a single certificate which uses fields 9 and 10 (marked "No" i.e. unused in Table 2.2.6, above);
- a sequence of certificates each providing key-material for the subsequent certificate in the path; this is the approach adopted here; the chains here are at most two elements long.

The CSA provides for multiple (virtual) CAs, one is selected for the user *U* and one for the VASP *V*. These two CAs, although logically separate, i.e. with distinct signature key pairs, will be implemented on the same hardware platform and using the same software. This will mean that the two CAs can never be active concurrently, although, since the CAs are off-line entities, this will not cause any practical difficulties. Also, because of the use of two distinct CAs, the certificate chains used in Protocol Variant C will not collapse to the single certificate case (except for CertChain(*V,T*)).

Definition of Certificate Chains for use in Protocol Variant C

$CertChain(U,V) = CrossCert_U(CA_V) \,||\, CertV = Cert_U(CA_V) \,||\, CertV$

> i.e. *CertChain(U,V)* is the concatenation of $Cert_U(CA_V)$ (the certificate of CA_V signed by CA_U) *and* *CertV* (the certificate of *V* signed by CA_V)

$CertChain(V,U) = CrossCert_V(CA_U) \,||\, CertU = Cert_V(CA_U) \,||\, CertU$

> i.e. *CertChain(V,U)* is the concatenation of $Cert_V(CA_U)$ (the certificate of CA_U signed by CA_V) and *CertU* (the certificate of *U* signed by CA_U)

$CertChain(V,T) = CrossCert_V(CA_U) = Cert_V(CA_U)$

> i.e. *CertChain(V,T)* is the certificate of CA_U signed by CA_V (since we assume here that $CA_U = T$)

where

- CA_X is the certification authority with which X is registered
- $Cert_X(Y)$ is the certificate signed by CA of *X* on P_Y
- P_X is the public "verification" key of the entity *X*

2.2.3.3 TTP Implementation

2.2.3.3.1 VASP - TTP communications interface

To use the on-line certificate service offered by the TTP, the VASP sends a message to the TTP specifying for what identification a certificate is needed. This message also contains the certificate of the VASP.

The TTP sends a message containing the required certificate or an error identifier in response.

The message from the VASP to the TTP has the value 0x02 for the id, and the data field contains the following BitStrings:

- g^u, which the VASP received from the user,
- *idU*, which the VASP received from the user,

- *CertV.*

The format of the PDU from the VASP to the TTP is as follows:

0x02	length	len(g^u)	g^u	len(*idU*)	*idU*	len(*CertV*)	*CertV*

The length field (16 bits), as mentioned above, contains the length of the complete message (including identity and length) in bytes as an unsigned integer while all the other length fields (32 bits each) indicate the length of the following BitStrings in bits represented as an unsigned integer.

The response from the TTP has the value 0x03 for the id, and the data field contains the following BitStrings:

- *TT*
- *CertChain(U, V)*
- *CertChain(V, U)*
- *CertChain(V, T)*
- $Sig_T(h3(g^u \,||\, cidU \,||\, cidV \,||\, TT))$

The format of the response PDU from the TTP to the VASP is as follows:

0x03	length	len(*TT*)	*TT*	len(*CertChain(U,V)*)	*CertChain(U,V)*	len(*CertChain(V,U)*)

CertChain(V,U)	len(*CertChain(V,T)*)	*CertChain(V,T)*	len(*Sig*)	$Sig_T(h3(g^u \,||\, cidU \,||\, cidV \,||\, TT))$

The usage of all the length fields is the same as in the previous PDU.

Time stamping

Time stamps will be encoded as the universal time type, UTCTime, the format used in the ASN.1 standard is used here.

The format of the time stamp is *YYMMDDHHMMSSZ* where *Z* may be either 0 (for Greenwich Mean Time) or a time differential.

Each pair of values (*YY, MM, DD, HH, MM, SS*) and *Z* have a length of 1 byte binary.

Field	Contents	Description	Size
1.	YY	Year	1 byte integer
2.	MM	Month	1 byte integer
3.	DD	Day	1 byte integer
4.	HH	Hour	1 byte integer
5.	MM	Minute	1 byte integer
6.	SS	Second	1 byte integer
7.	Z	Time Differential between local and Greenwich Mean Time	1 signed byte integer

Z ranges between -12 and 12 (and ignores the time zones with 30 minute offsets).

If *YY* is:

- greater than or equal to 50 it should be interpreted as 19*YY*.
- less than 50 it should be interpreted as 20*YY*.

Error Codes/Messages

Section 2.2.1.8 defines an error code format. TTP errors are defined as

id	**length**	**error_id**
FF	4 bytes	0 = not defined 1 = **user not known** 2 = not used (this could be used to mean "**old**" if the VASP does not itself check validity) FF = **revoked certificate**

2.2.3.3.2 TTPA functionality

- receive incoming request from VASP (message #02 in Protocol Variant C);
- analyse and check messages and certificates;
- retrieve relevant certificates from the Certificate Cache;
- assemble the response message and sign the message;
- send response message to VASP (message #03 in Protocol Variant C);
- User Interface;
- logging (tracer + monitoring).

2.2.3.3.3 CSA functionality

- receive user request for
  (a) secret/public key pair and certificate,

(b) certification of public key supplied by user,
(c) revocation of certificate;

- generate certificates;
- maintain certificates (and 'revocation list');
- store certificates in Certificate Cache;
- User Interface - GUI (plus possible use of diskette)
  - user requests,
  - user responses;
- logging (tracer + monitoring).

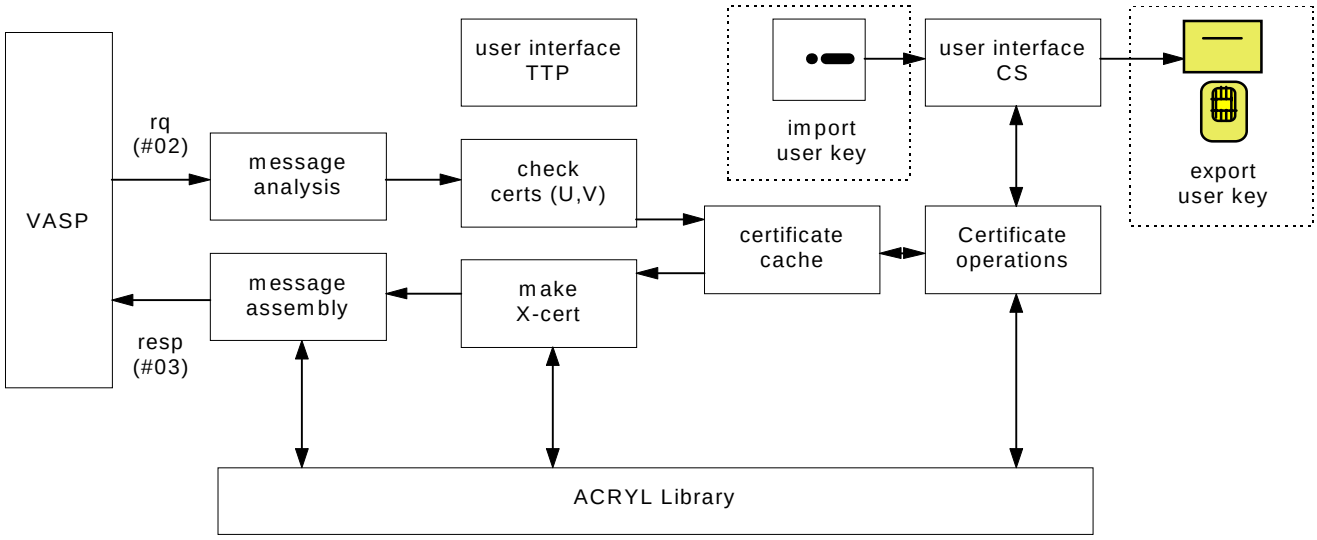

Figure 2.2.13 - Detailed architecture for TTP

2.2.3.3.4 Certificate Cache

The Certificate Cache is used for storing and retrieving the required certificates. The certificates are stored there by the CA (when they are generated after a user's request) and retrieved by the TTP whenever they are required for the Certificate Chains. Each certificate stored in the Cache carries a validity flag which indicates its status (whether it is active or it has been revoked). This flag is appended to the Certificate whenever this is issued, updated, or revoked. When the certificate is retrieved from the Cache by the TTP, its flag is checked and if it indicates a "bad" status, an error is signalled as output.

The format of each entry in the Certificate Cache is as follows:

Field	Contents	Description	Size
1.	Length	length of this record	1byte binary
2.	*idX*	identity of certificate subject in clear	2 bytes
3.	*cidX*	serial number (replicates Field 3 of certificate)	12 bytes binary

4.	*status*	`#00 - cert OK`
		`#01 - absent/not_created`
		`#02 - old`
		`#FF - cert revoked`
			1 byte binary

5.	*CertX*	Certificate on P_X (public key of X) as specified in Table 2.2.6	as specified for certificate (154 bytes binary)

Table 2.2.7 - Certificate Cache format

2.2.4 Plan

2.2.4.1 Milestones and Key Dates

	Date	Milestone Description
M01	06-APR-98	Trial specification available (this section of this document)
M02	08-APR-98	Trial documentation complete: operating instructions & user questionnaire
M03	20,21,23,24-APR-98	Conduct trial on EXODUS platform

2.2.4.2 Scope and Limitations

Users

The trial system supports only one user at a time; the objective is to obtain user opinion about the approach, and to be able to measure aspects of system performance characteristics.

2.2.4.3 Risks and Dependencies

2.2.4.3.1 Concerning Project EXODUS

See also *ASPeCT/EXODUS Memorandum of Understanding* [ASP/EX]

The dynamics of the relationship between the User, the VASP and the TTP service are dependent on the availability of simultaneous communications connections from the VASP to the User terminal and to the TTP server.

If simultaneous connections are not offered by EXODUS then the on-line connection to the TTP will not be demonstrated, and a fall-back will be shown, based on cached certificates held by the VASP.

It should be pointed out that it is the responsibility of EXODUS to provide the basic configuration as described above. This means:

- EXODUS must provide three terminals at one trial site connected with the EXODUS network.

- It must be possible to initiate ATM connections between the terminals without using the authentication trial smart card (even without the authentication smart card terminals attached).
- The ATM hardware driver on the user terminal and on the VASP terminal as well as the ATM switch must be configured to route Windows Sockets connections through the fixed ATM connection between the terminals (The ATM hardware driver maps the Windows Sockets connections onto a special fixed ATM connection and the ATM switch must be configured to provide this special fixed ATM connection with a reserved VCI/VPI address pair. With this configuration there is no problem having any two applications communicating via Windows Sockets, one on the user terminal and one on the VASP terminal).
- On the VASP terminal and on the TTP terminal the IWUs acting as 'gateways' for the ASPeCT applications (payment and TTP) must be installed and provide local Windows Sockets 1.1 communication to the ASPeCT applications.

There is no use trying to install the ASPeCT software on the EXODUS terminals before this configuration works It is suggested that this configuration is tested using commercial applications based on Windows Sockets 1.1, both for the communication between user and VASP PCs using the fixed ATM connection and between the VASP and TTP PCs via the IWUs. (Appropriate SW can be provided by Siemens if desired.) As soon as this configuration is working the TTP and secure billing smart card reader provided by ASPeCT can be attached to the user terminal and the ASPeCT software can be installed and tested on all three terminals.

2.2.4.3.2 Arising from ASPeCT

None.

2.3 Fraud Detection Trial

2.3.1 Technical Description

2.3.1.1 B-number analysis

In [D18] a detailed technical description of the functionality of the B-number analysis was given. In this section we address the issue of the integration of this tool with the other modules in the BRUTUS sequence as well as giving a brief summary of the techniques involved.

2.3.1.1.1 The need for a B-number analysis

The most common and expensive type of fraud, that of Subscription Fraud, will normally involve a subscriber using a false identity to purchase as many phones as he can in order to sell them, or the air time, to people wishing to make cheap international calls. Some destinations can be particularly identified with this kind of fraud. The purpose of the B-number analysis tool is to monitor the destinations of calls on a per subscriber basis, weighting the destinations of calls differently so that well known destinations for fraudulent calls can be given special attention. It is then down to a human operator, provided with such information, to decide whether or not to bar the subscriber.

2.3.1.1.2 Essential components for integration

For the trial implementation, the B-number analysis will be the first module that receives the toll tickets from the simulation of the billing mediation device (see section 2.3.3.1). The output from the module will be reformatted toll tickets that include tagging information to identify the source and nature of different fields. The six fields passed to the B-number analysis are the IMSI, Charging Start Date, Charging Start Time, Chargeable Duration, B-number and B-type of the call. The first task of the B-number analysis tool is to reformat these fields adding the tags that were agreed to identify each field. The tag identifiers are TMSI for IMSI, TCSD for charge start date, TCST for charge start time, TCDR for chargeable duration, TBNB for the B-number and TBTP for the B-type of number. To the end of this is appended the current alarm value for the subscriber in the BALM field. The B signifies to subsequent processes, receiving the modified Toll Tickets, that the B-number analysis tool generated the field.

2.3.1.1.3 Risk Analysis

In order to weight calls according to their country of destination, we group together countries belonging to the same geographical area or countries which we expect to have strong economic bonds. Indeed, it is assumed that people tend to have more contacts with people of neighbouring countries or, for business reasons, with their economic partners. This leads to the implementation of 10 different classes corresponding roughly to the following regions: North America, Africa, South America, Australia, Asia, Russia, Eastern Block, European Community, Middle East and Central Asia.

During runtime, for each Toll Ticket related to an international call, the country code is extracted. As the country code is kept unsanitized (see section 2.3.2.1), extracting it boils down to truncating the leading ‘F’ characters, included during pre-processing to pad out the TT-NON-CHARGED-PARTY field, and extracting two or three characters, depending on the country code, considered. Indeed, to make a correct classification, we sometimes have to distinguish between dialling codes such as 00-353, which belongs to Ireland and should be assigned to the European Community class, and 00-355 which belongs to Albania and should be assigned to the East-Block class.

2.3.1.1.3.1 Initial weightings for countries.

It would be anticipated that the weightings for different countries should be changed in the light of information being made available. For the purposes of the trial we have weighted calls in a way that is inversely proportional to the frequency that the zone in question is called, over a training population. This gives us the histogram as shown below where the scale used is logarithmic :

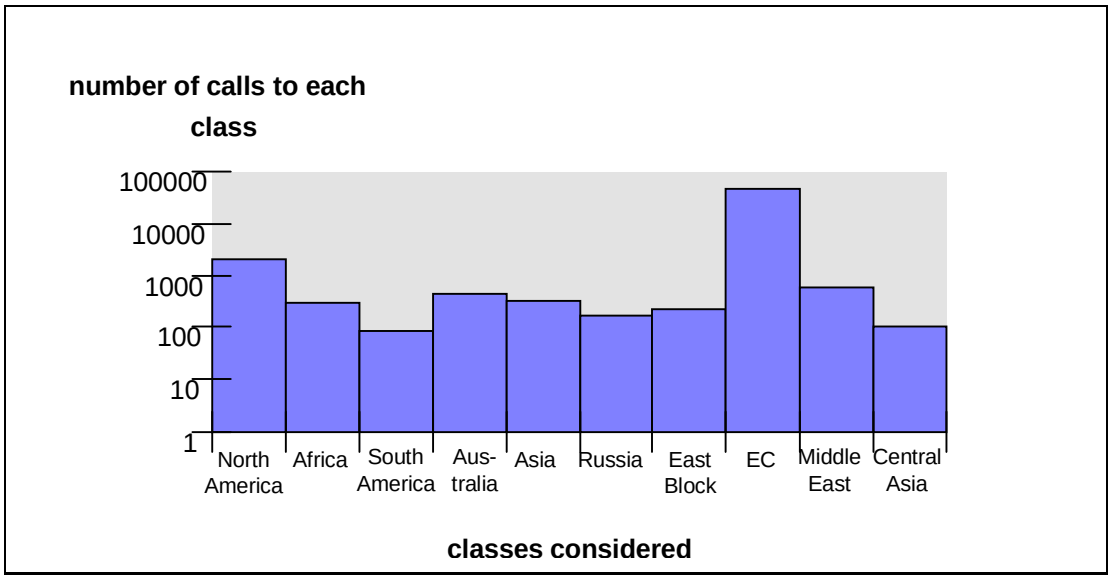

Figure 2.3.1 - Number of calls made to each class over the sequence of international Toll Tickets

It can be seen from this figure that the majority of calls are made to the European Community class. We decide thereupon to assign a weight of 0.5 to the European Community class, and to assign each other class a weight given by:

$w_i = 1 - \frac{n_i}{\sum_{j \neq k} n_j}$, where n_i is the number of calls to class i, and k is the index of the European Community class. The factor 0.5 for the European Community was chosen arbitrarily. It has to be high enough to allow changes in this class to raise an alarm and low enough so as not to raise an alarm too often.

2.3.1.1.4 Profiling B-numbers to produce alarm levels for the BALM field

As a Toll Ticket arrives for a user, the tool first determines if the call made concerns an international destination, and applies the international analysis if necessary. Each time an international call is made the Toll Ticket is assigned to one of the classes it belongs to and a vector of counters keeps track of the number of times each class has been excited by an incoming Toll Ticket. By considering two different time-spans over the toll tickets, we generate two profile records for each user. The profile representing the shorter Toll Ticket span represents the user's most recent activity and is called **CUP_int.** The longer span represents the user's history of usage and is called **UPH_int** The two profiles are maintained as probability distributions using two different decay factors α and β, both between 0 and 1. When a new Toll Ticket arrives, the user's **CUP_int** is updated. Each element of the **CUP_int** is multiplied by the factor α. The class to which the incoming Toll Ticket belongs is incremented by a factor $1-\alpha$.

The update rules for CUP_int are thus:

$$CUP\_\text{int}_{i\,new} = CUP\_\text{int}_{i_{old}} * \alpha \qquad \text{for } i \neq k$$

$$CUP\_\text{int}_{k_{new}} = CUP\_\text{int}_{k_{old}} * \alpha + (1-\alpha) \text{ for } i = k$$

with *k* being the number of the class to which the Toll Ticket belongs and *i* referring to the index of each class.

By assigning a 1 to the class which the Toll Ticket belongs to, the first time an international call is made, and using this updating technique, the profile is maintained as a probability distribution function. After updating the **CUP_int** both profiles are presented to the fraud engine that will determine the alarm level. It is necessary to allow both profiles to develop adequately for each user before considering the alarm level as evidence that anomalous behaviour is occurring. Following presentation to the fraud engine, the **UPH_int** is updated by incorporating information from the **CUP_int** to it and by using a decay factor of β.

The update rule used for the UPH_int is:

$$UPH\_\text{int}_{i_{new}} = UPH\_\text{int}_{i_{old}} * \beta + (1-\beta) * CUP\_\text{int}_{i},$$

where *i* refers to the index of each class.

The exact value of the two decay factors α and β is critical to the success or failure of the system and has still to be determined by experiment. So as not to increase unnecessarily the overhead on the system, we use the same α and β factors as the one used in the A-number analysis.

The fraud engine now proceeds by taking the B-number profile record consisting of the CUP_int and UPH_int as an input and calculates a *modified* Hellinger distance [D08] over all the weighted entries in the profile record. In this way we can attach more importance to changes occurring to classes that a genuine subscriber rarely calls while minimising the influence of changes to classes that are very often called. The Hellinger distance is then passed on to subsequent processes in the BALM field.

2.3.1.2 Unsupervised neural network tool

The unsupervised neural network tool is second in line in the BRUTUS sequence of modules. Its function is to build statistical behaviour profiles based on the classification of incoming Toll Tickets that uniformly span the piecewise continuous space of all possible Toll Tickets, to prototypical values in this space. By maintaining these profiles as probability distributions, we call upon classical statistical results to detect when, over a period of time, the distribution shifts significantly.

The strength of the unsupervised learning system, outlined in previous deliverables ([D08], [D13]), would be the ability to detect new fraud scenarios. In addition, alarm information from the unsupervised tool is passed on to subsequent modules as an alarm value in the UALM field. This value is then available to add information to subsequent analyses. The alarm value from the unsupervised tool indicates how erratic the subscriber's behaviour is.

2.3.1.2.1 Prototyping

Prototyping is a method of forming an optimal discrete representation of a naturally continuous random variable. The processing of continuous random variables by discrete systems generally reduces empirical information. Neural networks are capable of forming optimal discrete representations of continuous random variables through their ability to converge - by lateral interaction - to stable uniformly distributed states.

Grabec [Gra91] introduced a technique to dynamically generate prototypical values to span a continuous random variable as samples are taken from it. He also suggested an extension to generate prototypes for multi dimensional random variables. In its simplest form his method resembles the more well-known self organisation technique developed by Kohonen in 1988 [Koh88]. However, Grabec's method does not restrict

the prototypes to lie on a two dimensional manifold, but allows them to form their own topology. Only one pass through the training set is required giving rise to the potential for online adaptation.

Grabec introduced the second maximal entropy principle stating that:

*The mapping of a continuous random variable **X** into a set of K discrete prototypes **Q** reduces the empirical information by the least amount if a uniform distribution* $\{\mathbf{P}(q_i) = \frac{1}{K}, i = 1...K\}$*, corresponding to the absolute maximum (* $S_Q = \log K$*) of information entropy, is assigned to **Q**.*

When considering the set of all possible Toll Tickets, we clearly need a dimension to represent every parameter from a Toll Ticket which we wish to include in the analysis. Each parameter of a Toll Ticket can assume a range of values and is thus itself a random variable. Grabec's technique enables us to create a number of prototypes that dynamically and uniformly span the set of samples from a download of Toll Tickets taken from a live network. Owing to the fact that there are so few fraudulent Toll Tickets in comparison to non-fraudulent ones, in a live network download, the prototypes will organise themselves as if the data were totally fraud free. The resulting set of prototypes will enable us to classify future incoming Toll Tickets with minimal loss of empirical information.

To distribute the prototypes over the input stream of Toll Tickets, we set up an iterative procedure that computes the change in the current value of the K prototypes **Q**

$$\Delta q_{lm}^{(i+1)} = B_{lm} - \sum_{k \neq l}^{K} \sum_{i \neq m}^{M} C_{lmki} \Delta q_{ki} \qquad ; \qquad l = 1...K;\ m=1,...,M$$

which starts with $\Delta q_l^{(0)} = B_l$. The coefficients are determined by the expressions:

$$C_{lmki} = \left[\delta_{mi} - \frac{(q_{lm} - q_{km})(q_{li} - q_{ki})}{2\sigma^2} \right] \exp\left[\frac{-(q_l - q_k)^2}{4\sigma^2} \right]$$

where $\sigma \approx \frac{S}{K^{1/M}}$ approximates the standard deviation and S is the expected range of **X,** and

$$B_{lm} = \frac{K}{N+1} \left\{ (X_{N+1,m} - q_{lm}) \exp\left[\frac{-(X_{N+1} - q_l)^2}{4\sigma^2} \right] - \frac{1}{K} \sum_{k=1}^{K} (q_{km} - q_{lm}) \exp\left[\frac{-(q_k - q_l)^2}{4\sigma^2} \right] \right\}$$

2.3.1.2.2 Constructing profiles

Using the *K* Toll Ticket prototypes ***Q***, we can now encode future Toll Tickets as feature vectors ***v***

where $v_j = \frac{\exp(-\| X_{N+1} - Q_j \|)}{\sum_{j=1}^{K} \exp(-\| X_{N+1} - Q_j \|)}$.

Note that $\sum_{j=1}^{K} v_j = 1$ and so the feature vector can be viewed as a probability distribution.

Feature vectors are generated for National Calls, International Calls and for the use of supplementary services. The CUP and the UPH are now also formed as probability distributions using two different decay factors α

and β to maintain the concept of two time spans over the Toll Tickets. As the UPH is only updated after a fixed number of Toll Tickets has been processed, and thus a fixed number of updates to the CUP, we need to make α dependent on β.

To maintain the UPH as a probability distribution, we define its update rule to be

$$H_i = \beta H_i + (1-\beta)C_i$$

where C_i is the *ith* entry of the CUP. Note that $\sum H_i = 1$.

If we were to assume a similar style update for the CUP, after encoding each Toll Ticket into the feature vector ***v***, it would be performed in the following way

$$C_i = \alpha C_i + (1-\alpha)v_i$$

again ensuring that $\sum C_i = 1$.

We delay the update of the UPH until a batch of *B* Toll Tickets has been processed in the CUP. Because we are delaying the update of the UPH we need to make sure that, when it is updated, the influence of each Toll Ticket is as if it had been updated by decaying with β after every update of the CUP. In order to create this effect there will be a different decay factor for each Toll Ticket in the batch. We denote the decay factor for the *t*-th Toll Ticket in the batch by α_t. For example when processing the second Toll Ticket we need to set

$$\alpha_2(1-\alpha_1) = (1-\alpha_2)\beta .$$

This can be iterated to the *t*-th Toll Ticket giving the relationship

$$\alpha_{t+1} = \frac{\beta}{1-\alpha_t+\beta} .$$

This gives the iteration dependent update rule for the CUP as

$$C_i^{t+1} = \alpha_{t+1} C_i^t + (1-\alpha_{t+1})v_i$$

for the *t*-th update of the CUP modulo the block size *B*. Following this, both the CUP and UPH are presented to the fraud engine as discussed in the following section.

2.3.1.2.3 The unsupervised fraud engine

The task of the fraud engine is to take the user profile record, consisting of the CUP and the UPH, and calculate a measure known as the Hellinger distance

$$d = \sum_{l=0}^{K} \left(\sqrt{C_i} - \sqrt{H_i}\right)^2$$

where *C* and *H* are the CUP and UPH respectively and *K* is now the number of entries in the profile record. This is a natural measure to take when comparing two probability distributions. The Hellinger distance will always be a value between zero and two where zero is for equal distributions and two represents orthogonality. The Hellinger distance in this scenario can be seen as a measure of how erratic the behaviour is. The fraud engine then passes *d* on to subsequent modules in the UALM field.

2.3.1.3 Supervised neural network tool

The general architecture for the neural network approach to fraud detection is described in previous deliverables ([D06], [D08]) and we refer the reader to these documents for a first presentation of the fraud detection tool and of the real-time environment. We describe here the specifics of the implementation of the fraud detection tool. That is, how this tool extracts the User Profile Record, the Current User Profile, and the User Profile History from sequences of toll tickets; how it extracts the relevant features from these profiles; and how the neural network processes these features to produce its decision.

2.3.1.3.1 Profiling

2.3.1.3.1.1 User Profile Record (UPR)

The six fields that the mediation device simulator provides to the system are the TT_IMSI, TT_CHARGING_START_DATE, TT_CHARGING_START_TIME, TT_NON_CHARGED_PARTY, TT_B_TYPE_OF_NUMBER. This information is stored in the User Profile Record. However, we obtain the Current User Profile and User Profile History by using a filtering technique on the User Profile Record, we therefore need to translate the TT_CHARGING_START_DATE and TT_CHARGING_START_TIME to numerical values. The mediation device simulator converts thus the TT_CHARGING_START_DATE to the number of days from some reference date using a calendar and the TT_CHARGING_START_TIME to the number of seconds from midnight. This way, the absolute time of beginning of the call is equal to (in seconds from the reference date on midnight) TT_CHARGING_START_DATE * 86400 + TT_CHARGING_START_TIME.

2.3.1.3.1.2 Current User Profile (CUP)

The Current User Profile contains the information about the short-term behaviour of the user. We have decided to focus on the following measures of the behaviour of the user: the duration of calls and the interval between calls. High call duration, and low call interval (high call frequency) are indicators of, for example, call selling; while low call duration and low call interval are indicators of a PABX attack. We further split the call duration and the call interval between national and international calls. Furthermore, we do not only compute the mean duration of calls and the mean interval between calls, but also the variance of the call duration and of the call interval.

If we work with the absolute time of a call, we have the problem that calls tend to have very high intervals during the night, and low intervals during peak hours. To compensate for this, we determine the daily distribution of the calls and re-parameterize time so that the activity is equally distributed. The result is that the "time" difference between 10 p.m. and 6 a.m., may be equal to the difference between 10 a.m. and 10:30 a.m., because the amount of activity during the two periods is equal. The absolute time of the last national call *tn*, of the last international call *ti*, and of the last other call, have to be part of the profile to compute time differences. The short-term averages are computed using a first order filter. At each toll ticket *t*, we can compute the average $<x(t)>\alpha$ of a quantity *x(t)* over all previous toll tickets as follows:

$$< x(t) > \alpha = \begin{cases} (1-\alpha) < x(t-1) > \alpha + \alpha . x(t) & \text{if } x(t) \text{ is defined} \\ < x(t-1) > \alpha & \text{if } x(t) \text{ is not defined} \end{cases}$$

It is important to note that the quantity *x(t)* might not be defined at every toll ticket; for example, only one of duration of national call *dn*, duration of international call *di*, and duration of other call do will be defined at a time. The filter is, in fact, a first-order low-pass filter, and it gives thus an estimate of the mean *E(x)* of the signal *x(t)* (if the signal is a sequence of independently identically distributed random variables). Further, the

filter can track changes in the mean. So, the short-term average of the duration of national calls will be $<dn>_a$ in our notation. To compute a short-term standard deviation of a quantity *x(t)*, we can still use similar filters but on $x(t)^2$. We derive this from the definition of the variance as follows ($\hat{\mu}, \hat{\sigma}$ being estimates of the mean μ and standard deviation σ):

$$\mu = E(x)$$
$$\sigma^2 = E((x-\mu)^2) \overset{i.i.d.}{=} E(x^2) - (E(x))^2$$
$$\hat{\mu} = < x >_\alpha$$
$$\hat{\sigma} = \sqrt{\max(< x^2 >_\alpha - (< x >_\alpha)^2, 0)}$$

This finally gives the following 10 fields for the Current User Profile:

- Absolute time of last national call t_n
- Absolute time of last international call t_i
- Short-term average of the duration of national calls $<d_n>_\alpha$
- Short-term average of the duration of international calls $<d_i>_\alpha$
- Short-term average of the squared duration of national calls $<(d_n)^2>_\alpha$
- Short-term average of the squared duration of international calls $<(d_i)^2>_\alpha$
- Short-term average of the call interval between national calls $<i_n>_\alpha$
- Short-term average of the call interval between international calls $<i_i>_\alpha$
- Short-term average of the squared call interval between national calls $<(i_n)^2>_\alpha$
- Short-term average of the squared call interval between international calls $<(i_i)^2>_\alpha$

2.3.1.3.1.3 User Profile History (UPH)

We derive the User Profile History in a similar fashion by filtering the Current User Profile with a first-order filter with parameter β. This means that the User Profile History is, in fact, a second-order filter of the signals. It thus estimates average quantities, but on a longer time scale than the Current User Profile. Processing the call duration and call interval through a first-order filter to obtain the Current User Profile, and again through another first-order filter to obtain the User Profile History minimizes the memory requirements for the updates of the filter, and therefore minimizes the load on the database of user profiles. Furthermore, the difference between the User Profile History and the Current User Profile can be interpreted as a second-order band-pass filter on call duration and call interval. This means that it is affected neither by very short-term variations (let us say, between one call and the next) neither by very long-term variations (therefore allowing us to track long-term changes in the behaviour of the user). The difference between the User Profile History and the Current User Profile allows us to detect deviations from the normal behaviour of a user. The date of first call is also kept in the profile to determine at which point differential analysis becomes applicable (since, at the beginning, the User Profile History does not contain any relevant information). These considerations results in the following User Profile History (using the same notation as in the previous paragraph).

- Date of first call
- Long-term average duration of national calls $<<d_n>_\alpha>_\beta$
- Long-term average duration of international calls $<<d_i>_\alpha>_\beta$
- Long-term average squared duration of national calls $<<(d_n)^2>_\alpha>_\beta$
- Long-term average squared duration of international calls $<<(d_i)^2>_\alpha>_\beta$
- Long-term average call interval between national calls $<<i_n>_\alpha>_\beta$
- Long-term average call interval between international calls $<<i_i>_\alpha>_\beta$
- Long-term average squared call interval between national calls $<<(i_n)^2>_\alpha>_\beta$
- Long-term average squared call interval between international calls $<<(i_i)^2>_\alpha>_\beta$

2.3.1.3.2 Feature extraction

The features used by the classifier will not be the content of the Current User Profile and User Profile History directly, but estimates of means and standard deviations. The means are obtained directly at the output of the filters, but the standard deviations must be computed as $\hat{\sigma} = \sqrt{\max(< x^2 > -(< x >)^2, 0)}$, where the bracket denotes a first-order or a second-order filter. This results in the following vector of features, which is the input to the classifier.

- Number of days since first activity
- Short-term mean of the duration of national calls
- Short-term mean of the duration of international calls
- Short-term standard deviation of the duration of national calls
- Short-term standard deviation of the duration of international calls
- Short-term mean of the call interval between national calls
- Short-term mean of the call interval between international calls
- Short-term standard deviation of the call interval between national calls
- Short-term standard deviation of the call interval between international calls
- Long-term mean of the duration of national calls
- Long-term mean of the duration of international calls
- Long-term standard deviation of the duration of national calls
- Long-term standard deviation of the duration of international calls
- Long-term mean of the call interval between national calls
- Long-term mean of the call interval between international calls
- Long-term standard deviation of the call interval between national calls
- Long-term standard deviation of the call interval between international calls

2.3.1.3.3 Storing user profiles

User profiles must be swapped between disk and main memory each time a new toll ticket arrives at the fraud detection tool. The performance requirements are severe, as peak performance must exceed 30 toll tickets per second. We obtain such performance by using a simple, but optimised database tool called GDBM. The database is accessed by a key, which is the IMSI of the user and provides a content, which is the concatenation of the Current User Profile and User Profile History.

2.3.1.3.4 Supervised Learning

After designing the front-end, we must design the classifier. The front-end processes the toll tickets to produce sequences of user profiles; and then extracts the features needed by the classifier from these profiles. The classifier then maps a vector of features to an alarm value between 0 and 1 using a multilayer perceptron.

2.3.1.3.4.1 Multilayer perceptron

The neural network used in the fraud detection engine is a multilayer perceptron. It is defined as follows. The network is composed of elementary units called neurons. Each neuron produces at its output a simple non-linear transformation of its inputs depending on the value of the weights of the network:

$$y = \sigma(\sum_{i=1}^{n} w_i x_i + w_0), \quad \text{where } \sigma(z) = \tanh(z) \text{ or } \sigma(z) = \frac{1}{1+e^{-z}}.$$

The neurons are then arranged in a two-hidden-layer network with D inputs, H_1 hidden neurons in the first layer, H_2 hidden neurons in the second layer, and C outputs. The outputs z_m of the network can then be defined as

$$h_{1_k} = \sigma(\sum_{l=1}^{D} w_{kl} x_l + w_{k0})$$

$$h_{2_l} = \sigma(\sum_{k=1}^{H_1} v_{ik} h_{1_k} + v_{i0})$$

$$z_m = \sigma(\sum_{l=1}^{H_2} u_{lm} h_{2_l} + u_{l0})$$

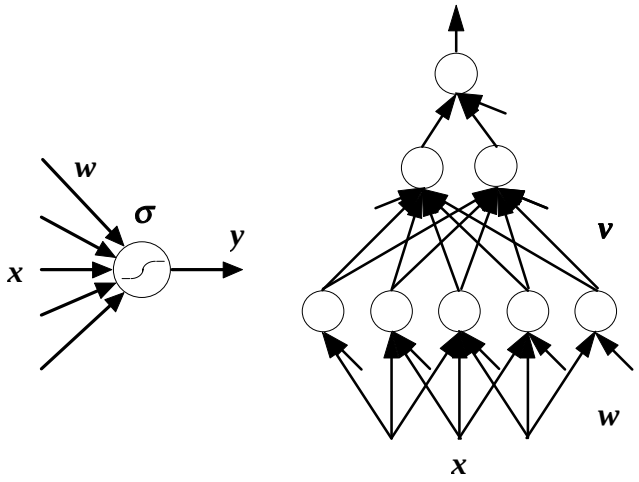

Figure 2.3.2 - Sigmoidal neuron and multilayer perceptron architecture

The main property of multilayer perceptrons is that they can approximate any function of the input to an arbitrary degree of accuracy, provided that enough hidden neurons are available. They can achieve this approximation with a relatively small number of parameters.

2.3.1.3.4.2 Labelling

For supervised learning, we organise the data available for design in a data set of labelled pairs $D = \{(X_1,Y_1),...,(X_K,Y_K)\}$, where Y_k is the fraud label ($Y_k = 0$ for normal behaviour, $Y_k = 1$ for fraud) associated to the k-th pattern with features X_k extracted from the user profile. The training data consists for the first part of the calls made by 300 users from the two-month download from Vodafone; these users are deemed normal. It also consists for a second part of the calls made by 300 fraudulent users. For all 600 users, all the available toll tickets are processed through the front-end of the system to produce sequences of user profiles. We label the sequences of user profiles for the normal users as non-fraudulent. For the fraudsters, we studied the evolution of the profiles over time to determine the beginning of the fraudulent behaviour; and we labelled the profiles as fraudulent during the fraudulent behaviour and as non-fraudulent otherwise.

2.3.1.3.4.3 Training

The first step is to choose the architecture of the neural network, that is the number of layers, and the number of neurons in each layer. Once we have chosen the architecture, the output of the network is a function of its input X_k and of the parameters w (the weights) of the neural network. There is a discrepancy between the output of the classifier $z(X_k,w)$ and the desired output Y_k. The learning of training of the classifier consists in adapting the weights so as to minimise this discrepancy. The measure of discrepancy is quadratic.

find w that minimizes $E = \sum_{k=1}^{K} \left\| Y_k - z(X_k, w) \right\|^2$.

We achieve this minimisation using a gradient-descent method, namely the Levenberg-Marquardt algorithm [Fle87]. This powerful method is based on algebraic procedures, which permits, given a value of the weights, determination of the modification of the weights that would lead to an optimal reduction of the error. After we have modified the weights, we have a smaller error and a new value of the weights. A new correction to the weights is evaluated, so as to reduce the error as rapidly as possible. We repeat this procedure until the error ceases to decrease.

2.3.1.3.4.4 Cross-validation

We split the data set into three subsets: the training set, the validation set, and the test set. In order to maximise the performance on previously unseen data we use the following procedure called cross-validation. The weights are adapted by minimising the error on the training set, but we observe the error on the validation set during this process; and we stop the minimisation when the error on the validation set reaches a minimum. We then estimate the expected performance on new data by computing the error on the test set.

2.3.1.3.4.5 Determination of the optimal architecture

We determine the optimal weights using the error minimisation procedure, but we have to repeat the procedure to search for a global optimum of the optimisation procedure, since gradient-descent methods are only guaranteed to converge to a local optimum. Furthermore, we have to repeat this procedures for different architectures of the neural network to determine the optimal one. Once we have found the optimal neural network, we simply have to use it on top of the front-end and it will produce an alarm value between 0 and 1 each time a toll ticket is presented to the fraud detection tool.

Within the trial, the supervised tool will also use the alarm level of the unsupervised tool as a form of a priori weighting in its training phase and later on in its alarm generation.

2.3.1.4 Rule-based tool

Basic ideas of the rule based approach, concepts and the architecture have been explained in [D06] and [D08]. After the evaluation of the first prototype in [D13], where the Rule-Based Tool has shown a high performance in detecting frauds, the final concept has been described in [D18], which is mainly an integration with the other fraud detection tools. Here in D19 we focus on how to realise the concepts of D18 and evaluate them in the final trial.

2.3.1.4.1 Determining an alarm level

One major precondition for the integration of several fraud engines is a unified measurement of fraud probabilities. Each of the fraud detection engines will compute an alarm level A with $0 \leq A \leq 1$ denoting a fraud probability. However, these alarm levels have to be standardised across all fraud engines so that they are comparable and can be used for ordering and common evaluations.

Within the first demonstrator the rule-based tool was using a common alarm level that we may name A_{demo} now, across all rules. For all rules i of the form: "if value V_i > threshold T_i, then raise an alarm" the common alarm level A_{demo} was defined as: $A_{demo} = \max (V_i / T_i)$

However, this value was a real number and not a fraud probability. For the integrated prototype we are mapping the alarm level A_{demo} to a fraud probability using the hyperbolic tangent: $A = \tanh (c\ A_{demol})$. A constant c will be used for approximation to the fraud probability of the other tools.

2.3.1.4.2 Applied rules within the integrated prototype

The rules applied have been and will be enhanced but still focus on the user behaviour determined by the most important Toll Ticket features as identified in prior documents (duration and number of national/international calls mainly). Since there are no cases of cloning in GSM known to the authors, overlapping call checks or velocity checks are not applied.

The rules are taking the following fraud indicators into account:

1. **High usage:** Most types of fraud are characterised by extraordinary high usage, the notable exception being personal use of fraudulent phones. The cumulative usage exceeds predefined absolute thresholds related to:
   - number of calls within a defined time interval
   - total duration of calls within a defined time interval

2. **Usage increase:** the recent usage is significantly higher than it was before. First, second or higher order differences can be built for evaluation. A user profiling techniques based on fixed time intervals as described in previous deliverables is used to support such differential analysis.

3. **Hot destinations:** this can be a single called destination, an organisation or a whole country. Hot destinations have previously been found to be involved in fraud. The integration with a B-number analysis tool allows now integrating such information into rules.

4. **Short calls (back-to-back calls)**: many subsequent calls of short duration can indicate hacking attempts for dialling through a PABX or free-phone service. A complex rule is responsible for recognising such patterns.

5. **Temporal factors** (time of day, day of week): A customer may usually not phone at certain times. A high usage at those times would be suspicious. Within companies, activities during out-of-office hours or during holidays can be an indicative of fraud. These facts will also be used for refining the rules.

2.3.1.4.3 Architecture of the rule-based tool

The architecture of the rule-based tool has only slightly to be changed and is shown in Figure 2.3.3. The main change is the introduction of a common stream between all fraud engines. For reasons described in [D18] the rule-based tool is placed best at the end of the sequence of all tools. The tool is now able to operate on toll tickets that have been attributed by the other tools and may be named ATTs (attributed TTs). This way, additional information provided by the other tools such as results of the B-number analysis can and will be used.

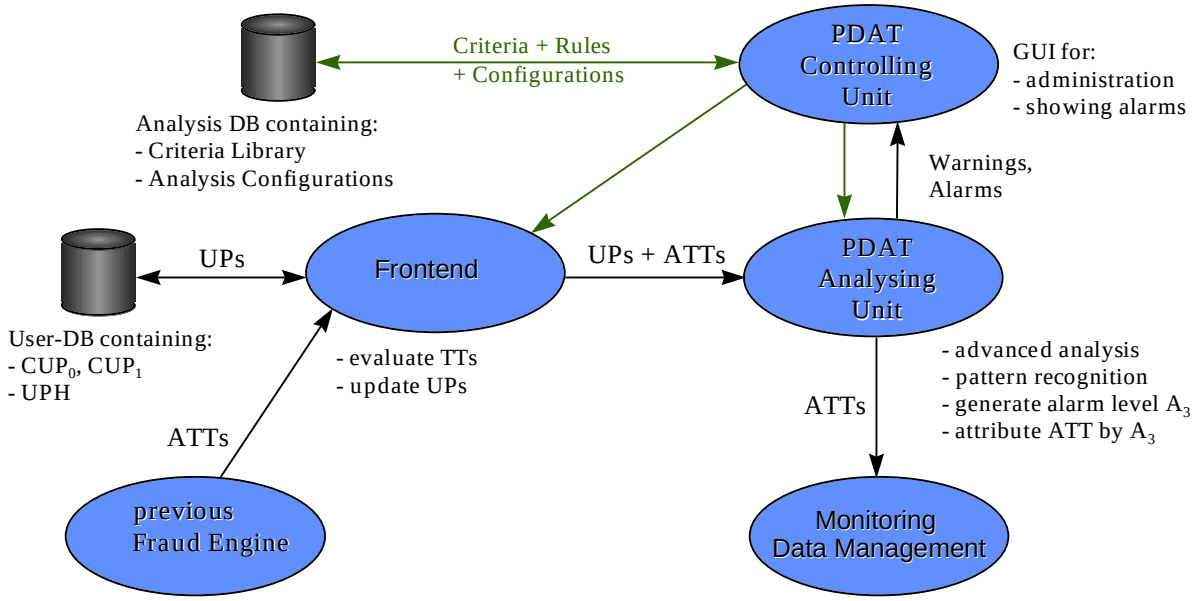

Figure 2.3.3 - Architecture of the rule-based part within the integrated prototype

As described in [D08], the Protocol Data Analysis Tool (PDAT) provides a comprehensive infrastructure based on a graphical user interface (GUI) for showing alarms and for editing alarm criteria during runtime.

Within the integrated prototype the part of the GUI showing alarms will be of less importance, since a more user friendly ranking of suspicious users is shown in the monitoring GUI now. A direct comparison with the other tools' results and an adjustable combination of these results will be shown as well.

The administration part of the PDAT GUI, however, is still the essential part for editing rules or changing complete rule configurations.

2.3.2 Trial Configuration and Methodology

2.3.2.1 Data sets

As input to the fraud detection trial, genuine network data need to be used. Suitable UMTS network data are not available because - apart from the fact that the number of users in UMTS trials is quite small - only simulated fraud could occur there. Therefore, the fraud detection concepts can not be validated in a UMTS system. Only in operational commercial networks, serving a large user population, is there a substantial probability that fraud attempts will occur. Thus, the Network Operators need to supply real usage data, representative of the operation of their network, not subjected to any ordering or filtering, to be used for the evaluation process.

Additionally, taking into account the data protection laws, the user data need to be converted in a way that prevents any user identification, before their processing by the fraud detection tools. This is the sanitization procedure, by which all fields with references to users' identities in the subscriber data are encrypted. The fields are enciphered without additional context, therefore allowing the comparison of these fields between different toll tickets. Thus, it can still be checked whether two fields refer to the same item (user or destination).

2.3.2.1.1 Panafon data

The data to be used in the trial are GSM toll tickets in an ASPeCT specific sub-set of the archived Eurobill format. This particular format contains 25 fields, among which the ASPeCT tools can isolate the particularly important ones for fraud detection. Examples of such fields are the A and B numbers (call originator and call recipient subscriber numbers), the call starting time, the duration of the call.

Such data are collected for a selection of 7659 users. The selection of those subscribers was performed during the first phase of the fraud tools development. It was based on the analysis of one week's toll tickets of all Panafon users. At first, the users were grouped in classes, depending on their level of usage of the network resources. Then, all high-usage subscribers were included in the selection, as this group presents the highest financial risk for the operator. Additionally, the selection included a sample group from every other usage category. It is therefore expected that the data from this selection will provide a variety of different and interesting behavioural patterns, representative of the Panafon users, to test the sensitivity and precision of the fraud detection components.

The sanitization code is then applied to the retrieved files. All subscriber information contained in the respective toll ticket fields is encrypted. Meanwhile, the 25 field format is preserved as well as all information that allows distinction between individual users. Thus, the confidentiality of personal data is protected while the case individuality is retained. In that final format, the sanitized toll tickets of active Panafon subscribers are stored on DATs and are ready for processing by the ASPeCT fraud detection tools.

This data is retrieved on a weekly basis from the Panafon archives. Each week retrieved produces three files. The ordering within each file is by subscriber. So far, six months of data for the selected subscribers are available starting from June 1997. Ongoing production of data up until the trials (September) is foreseen.

2.3.2.1.2 Vodafone data

Vodafone Ltd will provide a set of data containing approximately 6 months worth of toll tickets for approximately 20,000 users. The users have been chosen as a series of CHARGED-IMSI groupings, and it is therefore expected that a wide range of behaviours will be present within the data. The Toll Tickets are collected and stored on a daily basis. Once collected, this data will be concatenated into six 1 month files for distribution, training and processing. The size of the data set can clearly be further reduced by using only a subset of the users.

As with the Panafon, the data will be 'sanitized'. If any suspicious users are identified, the sanitization process can be reversed, and the user then investigated within Vodafone. This analysis will form part of the evaluation of the tools.

The Vodafone data will complement that provided by Panafon, by allowing the tool to be trialled against pseudo-live data obtained from a different network. It is expected that the fraud characteristics of the two networks will be quite different. This will also allow the flexibility of the tool to be assessed.

2.3.2.2 Trial Evaluation

The performance of the integrated tool will be evaluated in two senses. On the one hand, we will focus on numbers and percentages and measure the classification performance of the combined tool through the previously introduced method of the Receiver-Operator-Characteristic Curve. On the other hand, the usability of the tool's alarms will be investigated from the perspective of the Network Operators, each of those with its own network and way of managing data.

2.3.2.2.1 Evaluation of Technical Feasibility

To assess the performance of the combined fraud detection tool used in the trial, we will use the same, single performance index that was used to measure the performance of the individual tools in the previous reports. Also there we focused on finding a performance index that reflects the daily practice of fraud management. The striking feature of fraud detection is the importance of the trade-off between detection of fraudulent users and the production of false alarms. Indeed, we could develop a very conservative system that would generate alarms at the lowest levels of suspicion. But Network Operators and Service Providers are, from a commercial point of view, extremely cautious about unduly bothering good subscribers. Moreover, even levels of false alarms that would be considered excellent from a statistical point of view (let us say, one percent of misclassification), would be completely unacceptable in our case. For example, one-percent false alarms for one million subscribers means ten thousand false alarms. Conversely, we could guarantee that we do not generate any false alarms simply by not implementing any fraud detection system. Yet, the burden of loss of revenues caused by fraud makes this solution unattractive.

Therefore, the problem of the fraud detection tool will be to find the right balance between false alarms and correct detection. This optimal trade-off might be different for different operators, different services, or different periods. We thus developed the fraud detection tools to produce a single measure of suspicious behaviour each time it receives a new toll ticket. The decision itself comes from choosing an appropriate threshold and deciding to classify a user as suspicious if its activity rises above that threshold at any time of its profile history. A low threshold will guarantee high detection, but will generate many false alarms. High threshold will guarantee few false alarms, but will detect few fraudulent users.

The Receiver-Operating-Characteristic plots the percentage of correct detection of fraudulent users versus the percentage of false alarms for new users (as illustrated in the coming sections). The index of performance that we need to maximise is the surface under the curve. This is a very practical index of performance, more appropriate to our investigations than standard statistical measures. Such a trade-off curve will give the user of the fraud detection tools control over the fraud detection rate and false alarm rate.

2.3.2.2.2 Evaluation of User Acceptability

The users in this particular trial will be the two Network Operators, Panafon and Vodafone.

The user acceptability of the fraud tool will be evaluated in a number of ways.

Firstly, any suspicious behaviour reported from the trials against live data will be assessed, and additional work will be done to see if any of the subscribers are subsequently found to be fraudsters. Clearly there is only a little time available for such work between the conclusion of the trial and the final report. It is hoped that within months of data there exists some examples of fraudulent behaviour that can be detected by the tool, and subsequently be proved to exist by the networks within this little time available.

In addition to the above, the tool and its operation will be presented to the Operators' Fraud Teams, and its potential operation and use within the Networks discussed. The results of this discussion will be assessed and incorporated into the final report as an evaluation of the tools applicability to the Network Operators.

It is also believed that the fraud characteristics of the Panafon and Vodafone networks will be different. It will be a further interesting result of this trial to see if such a difference is detectable by the tool, and to see if the behaviour of the different networks can be characterised. This will also test the tools ability to adapt to different data, which is also relevant to the technical evaluation of the tool.

This will complete the assessment of the user acceptability of the fraud tool.

2.3.3 Realisation

2.3.3.1 Software architecture

In this section we present a high level description of the fraud detection trial configuration. The integrated tools will process Toll Tickets in a sequential manner as shown in the figure below. Toll Tickets flow through the architecture accumulating information pertaining to the analysis as it takes place. Subsequent modules have the ability to utilise this information in support of their own decisions. We refer the reader to the previous sections describing the technical approach to each of the modules.

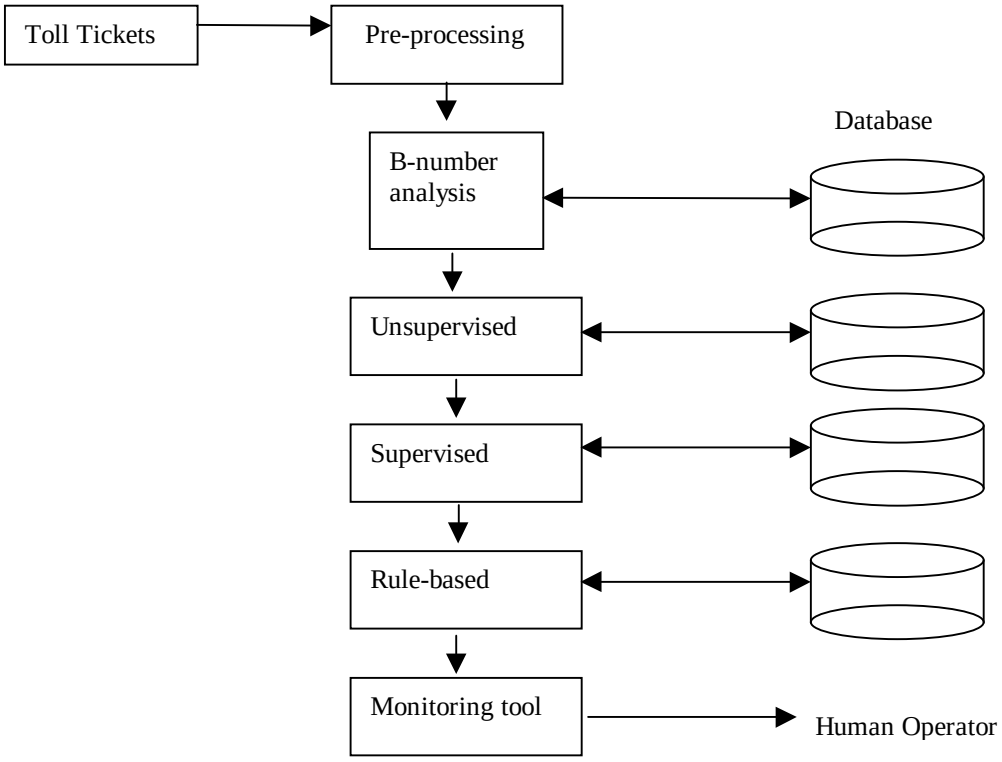

Figure 2.3.4 - Trial Architecture

All tools are using the same database implementation - which is GDBM, a simple and fast database on UNIX. This database fulfils the needs of user profiling, where the data records are always accessed via the IMSI as the only key. Within the first prototype the monitoring tool simply used a set of files for storing monitored users and the calls made by them.

Prior to the integration each tool was embedded into a common 'real' environment comprising a mediation device simulator feeding the tools with toll tickets and a monitoring tool to collect suspicious IMSIs and the corresponding call data. This environment will be kept for the project trial.

The trial fraud detection system will come with one common GUI to display the alarms. This GUI has mainly been finalised as part of the monitoring tool. In addition, some basic administration features are welcome enhancements. One of these features is a control-bar for each contributing tool to adjust their respective alarm thresholds. This way the tools are adjustable against each other during runtime and we can easily specify to what extent each tool contributes to the common result. A higher-level monitoring tool manages the exchange and representation of the information and for example also sets certain tuning parameters in the fraud detection tools for customising the fraud detection sensitivity.

The ordering of the fraud detection tools is motivated by the following considerations:

- The B-number analysis is added because it is believed that adding information on the destination of the calls will be able to boost the already reported performance of the individual tools. This module comes first because it adds information to the data-stream that could easily be used by all three tools.
- The Unsupervised NN is very good for novelty detection. It has good negative predictive value, which means that it can eliminate those users very easily for which certainly nothing is happening. Therefore it could be used as a first filter to all incoming calls. Another reason for putting this module high up in the chain is that its profiling could possibly also be used as input to the Supervised NN.

- The Supervised NN can efficiently pinpoint users whose behaviour is similar to previously observed and recorded fraudulent behaviour. Its training routines can be tuned to bias the performance towards a high positive predictive value, i.e. when it puts a fraudulent label on a user, the subsequent modules and/or human operator can be confident that there really is something happening.
- The Rule Based system does very well in explaining why alarms have been raised. It could for example be extended with extra rules to inspect why previous modules had raised an alarm. In this fashion, new fraud scenarios can possibly be identified. It can also be used to define hyper rules based on alarms raised by other tools and not only on its own profiling/information.

Initially, raw data from the network is pre-processed, discarding irrelevant components, and retaining useful data fields encoded in a suitable format. Secondly, the already present information on the user (i.e. the user's profile) is retrieved from the database. From the profile and the incoming data, relevant observables are derived. With these observables, we perform the following actions:

- The profile is updated and stored in the database for later re-use.
- An audit trail is maintained.
- The artificial intelligence component performs the fraud detection and generates a report on the alarm status of that user.

This report is then handled by the intelligent monitoring tool, which serves as a (graphical) interface to the human operator. Tasks performed by the monitoring tool are:

- Filter the types of alarm that the operator wishes to handle.
- Generate operator customised data and alarm level presentation for visual inspection.
- Set tuning parameters in the detection tools.

Raw data consists of TTs entering the prototype. The pre-processing block selects data fields and puts them into a format easily manageable by software. Each 'detection module' in the subsequent chain then extracts and uses the data of its choice from the general data-stream. Then the module adds its findings/results to the data stream while leaving the original data unchanged. The next module can then select from this augmented data-stream, the relevant information that it wishes to use in its own fraud detection. This means that subsequent fraud detection modules can use the profiling and/or the results of the preceding module. In a first implementation, where the work will be distributed over different computer platforms, each detection module will keep its own database.

Individual modules forward information that they receive from any other module and add tagged information of their own should it be required. The data is fully human-understandable at all times. It is structured as a sequence of tag/value elements.

- Tags are four-printable-symbol strings.
- Values are arbitrarily long printable-symbol strings.
- The size of a Tag label should be fixed.
- Blank spaces are used to separate tags and values.

The input behaviour of each tool is the following. When it reads a string, the tool scans it for the tag of the fields it wants to use and extracts the corresponding values, overlooking the tag/value pairs it does not use for its own processing. The first six elements (twelve fields) correspond to the six fields in the toll ticket used in the first demonstrator. The output behaviour is the following. The tool copies the string it received at its input directly to its output, and adds tag-value pairs for all the information it wants to output (for example, for use by the monitoring tool). Writing to standard output should only happen at one place in the code, so that the structure of the output can be updated easily.

Example:

The output of the Toll Ticket simulator might look as follows:

TMSI F23415124546303b2d224c63 TCSD 19960716 TCST 220038 TCDR 000693 TBNB
FFFFFFFFFFFFFF30198672b641014 TBTP 01 TSDN 0016 TSTS 079238 TBZC 03
TMSI F23415124546303b2d224c63 TCSD 19960716 TCST 221856 TCDR 000031 TBNB
FFFFFFFFFFFFF017433333d571a4f TBTP 00 TSDN 0016 TSTS 080336 TBZC 00
TMSI F23415124546303b2d224c63 TCSD 19960716 TCST 222015 TCDR 000367 TBNB
FFFFFFFFFFFFF017433333d571a4f TBTP 00 TSDN 0016 TSTS 080415 TBZC 00
TMSI F23415140807624d312f4b4c TCSD 19960716 TCST 224913 TCDR 000003 TBNB
FFFFFFFFFFFFFFFFFFF4646250c79 TBTP 00 TSDN 0016 TSTS 082153 TBZC 00

This output is sent to the fraud detection tool. But the tool might not need all these fields. Let us say that instead of using the B-type, it prefers to use the zone code TBZC (Toll Ticket B-number Zone Code) that gives the region of the world the call was made to. And it does not use the TSDN (Toll Ticket Starting Date Normalised) and TSTS (Toll Ticket Starting Time in Seconds) fields. It further processes the information, possibly producing an alarm. At the output, it copies what it received at the input plus all the information it finds relevant (here, the information about the alarms). The output could look as follows:

TMSI F23415124546303b2d224c63 TCSD 19960716 TCST 220038 TCDR 000693 TBNB
FFFFFFFFFFFFFF30198672b641014 TBTP 01 TSDN 0016 TSTS 079238 TBZC 03 SALR
ALARM SALV 0.87
TMSI F23415124546303b2d224c63 TCSD 19960716 TCST 221856 TCDR 000031 TBNB
FFFFFFFFFFFFF017433333d571a4f TBTP 00 TSDN 0016 TSTS 080336 TBZC 00 SALR
NOALR SALV 0.37
TMSI F23415124546303b2d224c63 TCSD 19960716 TCST 222015 TCDR 000367 TBNB
FFFFFFFFFFFFF017433333d571a4f TBTP 00 TSDN 0016 TSTS 080415 TBZC 00 SALR
NOALR SALV 0.33
TMSI F23415140807624d312f4b4c TCSD 19960716 TCST 224913 TCDR 000003 TBNB
FFFFFFFFFFFFFFFFFFF4646250c79 TBTP 00 TSDN 0016 TSTS 082153 TBZC 00 SALR
NOALR SALV 0.12
TMSI F23415137f381c3f526f710a TCSD 19960717 TCST 074916 TCDR 000001 TBNB

FFFFFFFFFFFFFFFFFF4646250c79 TBTP 00 TSDN 0017 TSTS 028156 TBZC 00 SALR ALARM SALV 0.98

TMSI F2341513363e667947231933 TCSD 19960717 TCST 080242 TCDR 000023 TBNB FFFFFFFFFFFFFF301423d5c494b49 TBTP 01 TSDN 0017 TSTS 028962 TBZC 03 SALR NOALR SALV 0.07

The added information consists of the flag SALR (Supervised NN ALaRm) and the alarm level SALV (Supervised NN Alarm LeVel).

2.3.3.2 Monitoring and Graphical User Interface

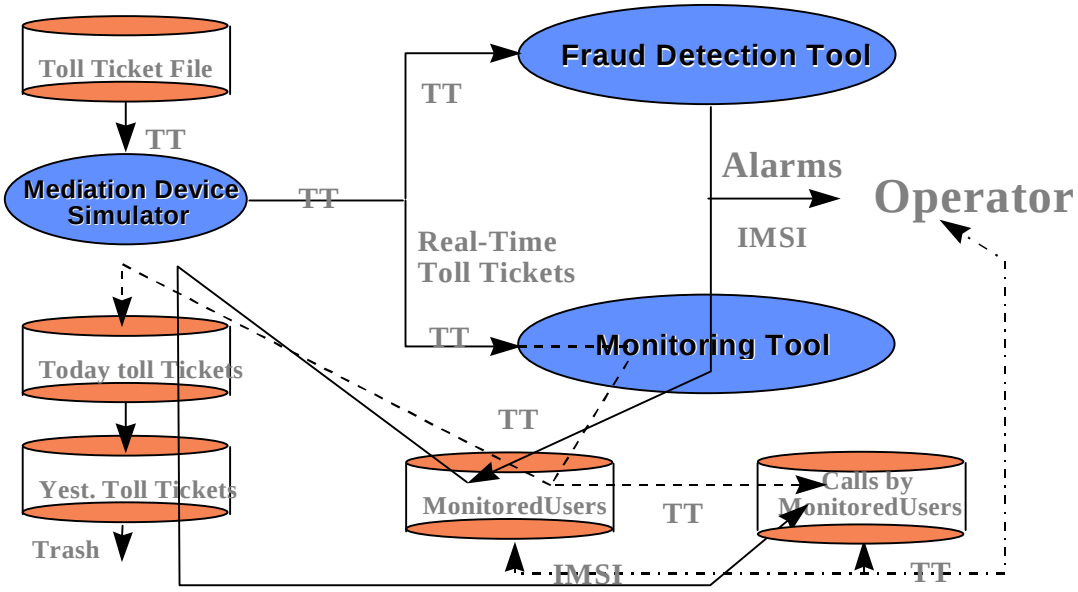

Figure 2.3.5 - Monitoring within the First Demonstrator

Figure 2.3.5 shows the monitoring as it was in the first demonstrator. The framework comprises a number of C, C++ and Perl software simulations to emulate a real time processing environment. Firstly, the billing mediation device is simulated to send Toll Tickets out to the fraud detection tool with a pre-set mean interval between each ticket and variance based on a Poisson distribution. These Toll Tickets are also passed to a monitoring tool, which checks for alarms being raised by the fraud detection tool. The monitoring tool then stores toll tickets for any subscribers exhibiting suspicious behaviour. The monitoring tool also keeps files containing the current day's and previous day's Toll Tickets in files sorted by IMSI. Once a subscriber has become suspicious his TTs are retrieved from these files and stored with the calls by monitored users.

For the integrated fraud system the monitoring framework will be improved in three ways.

Improvement 1: *Use fraud probabilities.* Instead of the information "alarm yes/no" we are using alarm levels A, $0 \le A \le 1$ denoting fraud probabilities. Each of the fraud detection engines will compute an alarm level and

will add this information to the common stream of information passed through all fraud engines towards the monitoring tool. We can view the TTs as being attributed with a list of alarm levels. These attributed TTs (ATTs) will be the basic information within the monitoring framework. The alarm levels have to be standardised across all fraud engines so that they are comparable and can be used for ordering and common evaluations. A framework of this monitoring is depicted in Figure 2.3.6.

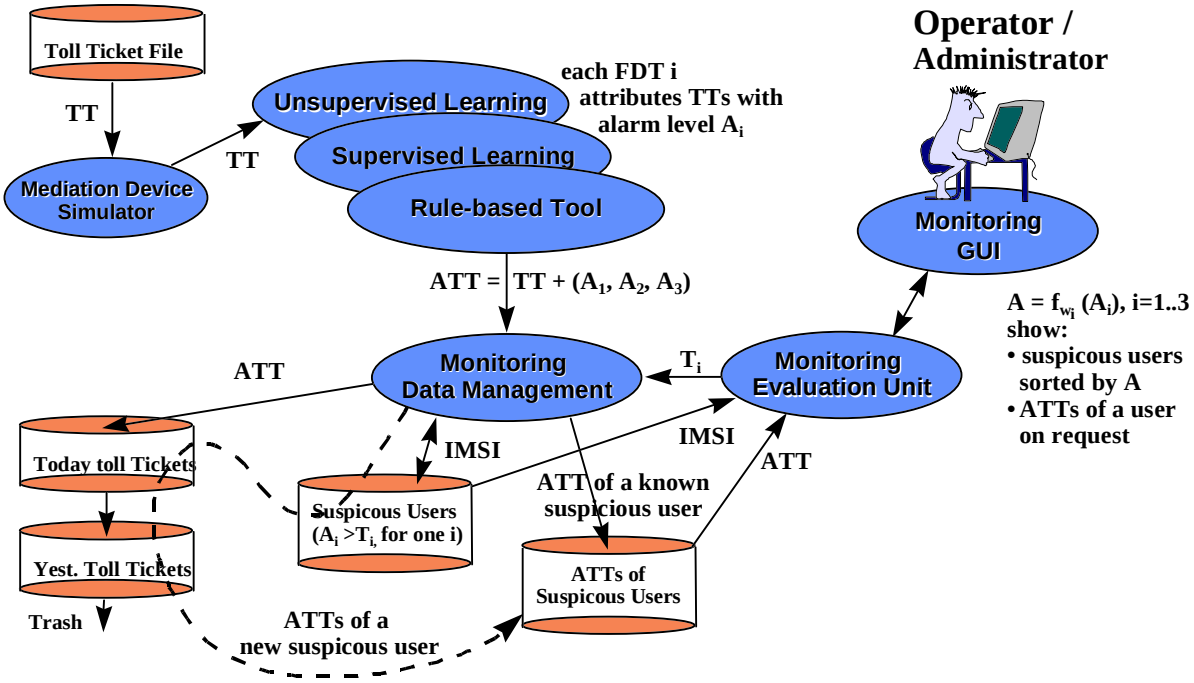

Figure 2.3.6 - Monitoring within the Integrated Fraud System

Improvement 2: *Evaluate The Monitored Information.* In the first demonstrator the monitoring task was restricted to data collection only. The result was a list of suspicious users and a list of all TTs of suspicious users. This part will be a basic part of the integrated prototype as well and is named "Monitoring Data Management". The main difference from the first demonstrator's monitoring is that we are using much lower thresholds, i.e. the subset of suspicious users for closer monitoring will be much larger than the set of monitored users in the first demonstrator. For each fraud engine a basic threshold T_i will determine whether a user will be classified as suspicious. A user will be classified suspicious if his alarm level exceeds one of the thresholds T_i. In general, a suspicious user is not a person raising an alarm. Suspicious in this case means candidate for further investigations. A user who is classified not suspicious at a certain time may however be classified suspicious later on.

When a user is classified suspicious for the first time, he will be stored into the suspicious user's file. All ATTs relating to him will be extracted from the buffers containing all TTs of the last two days. These buffers may also be expanded to a longer-term history. Subsequently, the extracted ATTs are stored in a separate file for further investigation. For users already known to be suspicious all ATTs are directly fed into the ATT file.

Of course, there must also be a mechanism to classify users as not suspicious again. This will be regularly done by inspecting the ATT-file of suspicious users. Such users who fall below certain thresholds will then be deleted from the suspicious users file.

Based on the suspicious users file and the ATT-file of suspicious users, an evaluation will be performed by the "Monitoring Evaluation Unit". The evaluation unit will work on the suspicious users and their ATTs only. The main evaluation task is preparing information about suspicious users as requested by the monitoring GUI. Thus, the evaluation unit provides a GUI-server. It is intended to implement the evaluation unit in Java. In view of the time limits in ASPeCT it will be checked whether it is possible to realise this server as a Web-Server accessible via HTTP.

Improvement 3: *Show suspicious users and behaviour via the GUI.* The monitoring GUI comprises functionality for supervision and for configuration. The most important feature will be showing suspicious users sorted by their alarm levels i.e. by the probability of being fraudsters. The related graphical element will be a table showing the fraudsters' IMSIs, the alarm levels A_i for each tool and a common alarm level A_{com}. By selecting a user entry in the GUI the set of ATTs relating to the user can be recalled.

The common alarm level $A_{com} = f\ (w_1, w_2, w_3, A_1, A_2, A_3)$ will be used for the ordering. The function f is used for combining the thresholds computed by all tools to a common alarm level. The weights w_i will allow to manually adjust the influence of each tool for the common result. It will be one main task of the trial to determine a well-suited function for combining the results.

Candidates for this function may be

$f = \max w_i A_i$,

$f = \Sigma\ w_i A_i$,

$f = \Sigma\ w_i A_i^2$ or

$f = \sqrt[3]{\prod(w_i A_i)}$.

Changing the weights will clearly be a task of an administrator and not of a normal operator. The same holds for changing the thresholds T_i for a minimum suspicion. An adjustable global threshold T_{com} will finally allow the raising of an alarm, if A_{com} exceeds T_{com}. This is meant for the critical cases where the tool should alert an operator and in a later stage proactively propose countermeasures.

If the time for development allows the implementation of a Web-Server, then any Web-Browser such as Netscape could be used as a monitoring GUI. This option would allow remote monitoring via the HTTP protocol independent of the computer platform. Otherwise, the GUI will be realised by a Java application window on the same computer as the evaluation unit.

2.3.4 Plan

This trial is not dependent on the EXODUS project or platform, or indeed any external project. The timescales and resources do therefore not have the same sort of risk or dependency as the trials described in 2.1 and 2.2, above.

2.3.4.1 Milestones and Key Dates

Reference	Date	Milestone Description
MF1	30-JUN-98	component tools ready for integration
MF2	31-AUG-98	integration complete
MF3	31-OCT-98	trial complete
MF4	31-DEC-98	trial analysis and final report complete

2.3.4.2 Risks, Limitations and Dependencies

There are no identifiable external risks or dependencies.

Annex 1

Below, the Authentication Trial data is presented in a number of tables. The additional abbreviations that are used, are:

- ID : identity
- IT : Italy data
- PK : public key

  NO: public key agreement key

  CA: public signature key

  User: public signature key
- SK : secret key
- SW : Switzerland data

The Basel set of UIMs have an IMUI between 228010200 and 228010219. They belong to the Swiss Service Provider 22801, and their certificates are issued by CA 3210.

The Milan set of UIMs have an IMUI between 222010200 and 222010204. They belong to the Italian Service Provider 22201, and their certificates are issued by CA 3211.

The Turin set of UIMs have an IMUI between 222010205 and 222010209. They belong to the Italian Service Provider 22201, and their certificates are issued by CA 3211.

Values that are prefixed with an apostrophe (e.g. `'D8C8817596B6278DD626704ED7A89412`) are a hexadecimal representation of binary data.

A1.Certification Authority data

	ID	PK	SK
SW	3210	'00000002010000008482048 33C6CFE89 5F7530798977FF23EE900100000084E3 CAAD4E1E7C528E06F1C2D799662B1890 OCTET STRING : 48 bytes	'D8C8817596B6278DD626704ED7A89412 OCTET STRING : 16 bytes
IT	3211	'00000002010000008 4C78D5D12E5AA19 D0BF6968EFD502F012B00100000084D6 9E6FBF09B71039AC60FF6A9DC77E8630 OCTET STRING : 48 bytes	'BFBDADBC44D0C737C9AA02005C852D05 OCTET STRING : 16 bytes

A2.Network Operator data

	ID	PK	SK
SW	2	'00000002010000084C740C1C23C742C 754732988CA019B6FB000100000084F1 AD19E452C074078A080C09E522DBFBE0 OCTET STRING : 48 bytes	'0D7DB5BA3E617818E3E030596590348C OCTET STRING : 16 bytes
IT	1	'00000002010000083EB26ED2AD23D1F 17DCD29643849E0222C00100000082D9 9AC11CA3F143544284F7A9812BD2BB80 OCTET STRING : 48 bytes	'2FB481BFB1660F81971353962A41E3A1 OCTET STRING : 16 bytes

ID	CA	Certificate
2	3210	'010300B8100000000000000000000000013332313000320 04E6574776F726B20 4F00000130B90B00000130E01B32004E6574776F726B204F7065726174020100 84C740C1C23C742C754732988CA019B6FB0F1AD19E452C074078A080C09E522D BFBE010081817B6B7C25E284E3F18AFBC0F90015183260B701EA6B7ED6A9EF8F 8F09851D1900 OCTET STRING : 134 bytes
2	3211	'010300B8100000000000000000000000013332313100320 04E6574776F726B20 4F00000130B90B00000130E01B32004E6574776F726B204F7065726174020100 84C740C1C23C742C754732988CA019B6FB0F1AD19E452C074078A080C09E522D BFBE01008145B595ED2567840511457085D8DC488F48F0A6E291C01E04482DFD BE95D63C8F40 OCTET STRING : 134 bytes
1	3210	'0102FEB8100000000000000000000000013332313000310 04E6574776F726B20 4F00000130B90B00000130E01B31004E6574776F726B204F7065726174020100 83EB26ED2AD23D1F17DCD29643849E0222CD99AC11CA3F143544284F7A9812BD 2BB8040201E5638B16295A5612FA9CC606CEC58F534B601C8CCD533802C6523F B646566A68 OCTET STRING : 133 bytes
1	3211	'0102FEB8100000000000000000000000013332313100310 04E6574776F726B20 4F00000130B90B00000130E01B31004E6574776F726B204F7065726174020100 83EB26ED2AD23D1F17DCD29643849E0222CD99AC11CA3F143544284F7A9812BD 2BB80402053AFA7400880F3390D15F6E749A5D795F5DE945FBAEC461B7BD0074 7B4A663B56 OCTET STRING : 133 bytes

A3.Service Provider data

Service Provider Switzerland: 22801

Service Provider Italy : 22201

A4.User data

A4.1.keys

IMUI	PIN	PK	SK
228010200	16838	'0000000201000000083D52259134FC926 572DE4A7EA1DA6BCF1A00100000084D0 E27D75707222576E829797C4D9665B10 OCTET STRING : 48 bytes	'C4FE66AE96AF790A73A6B61B50D22321 OCTET STRING : 16 bytes
228010201	5758	'0000000201000000084E7EBDDACDC573 3 E4FFC0A2F5870476A15001000000849E 2FEF4B007FBBF37271DEF6187EC380C0 OCTET STRING : 48 bytes	'A9CA4E4ED156DAF6C55C214B6B7FC5F F OCTET STRING : 16 bytes
228010202	10113	'0000000201000000084A8A0E1FC8F666 D BE6CCE351906DC79E2A0010000008382 89FF1E89735795141901CDB14485D340 OCTET STRING : 48 bytes	'A1B84A30200DF39F3F79827094A77C OCTET STRING : 15 bytes
228010203	17515	'0000000201000000083F1A621AD1B456 F 10BD44A5136A837B234001000000848B DE177E657DB2D29B8A4B50B8C1EFDB10 OCTET STRING : 48 bytes	'8BE5F61B7031CE2970FDA175D2F57CB A OCTET STRING : 16 bytes
228010204	31051	'0000000201000000084D14099B66E45B F 3FD91D6804D1CAEE28000100000084AD 7D4D2BD23D0A4DEE16A384D05F3C2590 OCTET STRING : 48 bytes	'C3ED3DB8D704CC06755832692C00234 C OCTET STRING : 16 bytes
228010205	5627	'0000000201000000084939586B87166A 2 D6B74483271891BC9EC0010000008484 3CB16196D52743914375ED3ED5A5A010 OCTET STRING : 48 bytes	'F488E809B56EA867FA5C6730DAED9BF 1 OCTET STRING : 16 bytes
228010206	23010	'0000000201000000084C3C461FB53C3C F 3B256ED262A7F1FAC0400100000084A6 119BC660DA3218847108BA2F2903F400	'BBBDF605E9E58A891E6B4499A98156A 9 OCTET STRING : 16 bytes

		OCTET STRING : 48 bytes	
228010207	16212	'00000002010000084A47C4B7677038F B8283DF81A46020180100100000082F7 BF17BD749BFA0381E1F144FC86ABFBC0 OCTET STRING : 48 bytes	'E75282696455E012EF9D13DFFDA79140 OCTET STRING : 16 bytes
228010208	7419	'00000002010000084DC010FA1F11554 2BB5AA6FEF78F00883B0010000008483 42CA0152B20B9FCF172577EDC581DA80 OCTET STRING : 48 bytes	'B5C7708DBFC600E6750C957B0FEADE2E OCTET STRING : 16 bytes
228010209	4086	'00000002010000084C66588002967B5 09A3803904C6DD81C9500100000083B7 8719F5CBF0BA182677DD55ACD2CBD5C0 OCTET STRING : 48 bytes	'E90F47C5A5CC163E83404E7A68AC6B05 OCTET STRING : 16 bytes
228010210	2749	'00000002010000084B321171B105969 CFBBA889CF7C287A9BA00100000083D5 7E5604A863AC256911B12A412C38B6C0 OCTET STRING : 48 bytes	'E1758F0CA447F57433B3529CA0B80128 OCTET STRING : 16 bytes
228010211	12767	'00000002010000084D6F4F6326D8C35 23E90CE5187ED232CE70010000008489 9B8F9198BF5067A53DC5EBFC6F28B6F0 OCTET STRING : 48 bytes	'AAF0634EF7E6EBB2C8212C8AC80EF765 OCTET STRING : 16 bytes
228010212	9048	'00000002010000084F6C23391F85320 4DFDEDA69F783C0A7D500100000084CF 88428A56EEE0BE78F6065907A4C518F0 OCTET STRING : 48 bytes	'BC8E3B0B9C149F0AA6706F9E2326E187 OCTET STRING : 16 bytes
228010213	12060	'00000002010000080BE004CD22C42C3 9A4801DB3213CD54DD0100000084C430 19F352932D0322F4425968FB5879B0 OCTET STRING : 47 bytes	'C89337A9868A5C7728A7AF87E3CC8706 OCTET STRING : 16 bytes
228010214	32225	'00000002010000084C7BFC3F4901ABF	'F79ABA821436C55A3AF3F3B001BA2519 OCTET STRING : 16 bytes

		E475C7E8D2BBDF5A39500100000084FB 12C5F5E0DD233EAD339E217F5D4A98B0 OCTET STRING : 48 bytes	
228010215	17543	'0000000201000000084A4E5A26394C6EB 4BB0B05E5DC0EDBB7F10010000007D9A 01247D4AFB24198CEE8226785D0D70 OCTET STRING : 47 bytes	'F62B1C247251B419C3A01E0C11A5188A OCTET STRING : 16 bytes
228010216	25089	'0000000201000000084C0D6A39677DDCE C13545232715F54B0B400100000083AB E75B529A30509135D01C388F69678AC0 OCTET STRING : 48 bytes	'A427CAA7943959F6F5C48945A1BE59F6 OCTET STRING : 16 bytes
228010217	21183	'0000000201000000081B92807644708F9 F4FA37127FCBC4E886800100000082EC E57EF624E6E0CD4BF8EB20D42C71C6C0 OCTET STRING : 48 bytes	'B997A9CF14513D65A1469A783FEBF0BA OCTET STRING : 16 bytes
228010218	25137	'0000000201000000083C6CC81E1EF64B2 2283C89A0C4658FBBEE00100000083C7 44C64E4126ECF716482EC9EF7E890F60 OCTET STRING : 48 bytes	'949242754953BB22D9744AC494902F14 OCTET STRING : 16 bytes
228010219	25566	'0000000201000000083F026A8E3122972 7B53C59E4296B4D737000100000084E1 2668582EE0FA56AADF4416B24C5B4110 OCTET STRING : 48 bytes	'C1D2B6723B97E9759E881A26B0105A2D OCTET STRING : 16 bytes
222010100	26966	'0000000201000000084B58DF7CD47EEC3 10290DFE3BB693AACC100100000084CF 90CBA38C2E328FBAC5CDC4FB50A78100 OCTET STRING : 48 bytes	'F008F50445E02DEBBA4947577A991C24 OCTET STRING : 16 bytes
222010101	4978	'0000000201000000081D3E4A95CE2D880 7ACE589570CB3641AE000100000084C7 1D8D36801A07D244671A06DDD78782E0 OCTET STRING : 48 bytes	'BFD13812D010B9FD94B0CA8CCFB71749 OCTET STRING : 16 bytes

222010102	10311	'00000002010000007E82A5BBF4F0D09F DC2D63B290A00C2C900100000084DF2E 2406FB944B7B72325EDD598BA7B850 OCTET STRING : 47 bytes	'C3F09D82AEA6FCF037659C0BF4214420 OCTET STRING : 16 bytes
222010103	20495	'000000020100000083849DE1F5B33763 CCBA2A1E52CA64FB59400100000082EC 904F2DF66C62E32D5F569589C2261580 OCTET STRING : 48 bytes	'9116502B0774F30C5A40395232B7EC84 OCTET STRING : 16 bytes
222010104	30054	'000000020100000084954FBA171F4154 D5A5A23C76631A1AF9D00100000081D8 BC3A93DF06201D455115AEBD086C9500 OCTET STRING : 48 bytes	'FF61A7A2EABB9C5FEB1CCE705DF354B6 OCTET STRING : 16 bytes
222010105	17031	'00000002010000083DFF81B8469FE12 A07C9D05B42712F37E000100000082DF F90470688577C697C34665FB7E9102C0 OCTET STRING : 48 bytes	'E1F31E42E267445DA404032D426D62CC OCTET STRING : 16 bytes
222010106	13145	'00000002010000083C5B5B5517B5985 34955FF9185CA5D07AA00100000083C3 6447FC6B46CB3851EBEA05170758FC60 OCTET STRING : 48 bytes	'B177FE74BDCB7A62CF916992CC6C6A14 OCTET STRING : 16 bytes
222010107	19882	'00000002010000084B2AE38E3EB571C 0C7167AD958071283FE0010000007FB6 0163B1239CA23028A6BD0F3365D4B6 OCTET STRING : 47 bytes	'E55575F5BEA7D98978AEA175F39FECCD OCTET STRING : 16 bytes
222010108	25763	'000000020100000848C9127E5E5B561 A1441A52679099C516200100000084F9 8DEC8D7716FB9D0328D89C26D86B9A20 OCTET STRING : 48 bytes	'AC03021D4CE82DEE169D74F6A3167FB9 OCTET STRING : 16 bytes
222010109	6561	'00000002010000083A78BCE53BD6F9B 3A84C899A2DE6F13DDA00100000082C4 0941177B325F9637CDFEDC003F4EB400	'E2AE6DFBB0D6936731951CE68D23A278 OCTET STRING : 16 bytes

	OCTET STRING : 48 bytes

A4.2.Certificates

IMUI	PK
228010200	'010300B810000000000000000000000013332313000557365722043657274696600000130B90B00000130E01B32323830313032303000323238303130020100846A912C89A7E4932B96F253F50ED35E78DD0E27D75707222576E829797C4D9665B10100818FC9636BDB3E7DED71BBA9E8E2E5CE59166D28F284077874F1E0CF56360EC7C9C0 OCTET STRING : 134 bytes
228010201	'010300B810000000000000000000000013332313000557365722043657274696600000130B90B00000130E01B32323830313032303100323238303130020100 84E7EBDDACDC5733E4FFC0A2F5870476A159E2FEF4B007FBBF37271DEF6187EC 380C010080F53F98DAC915E66F8B4063E5715C64E2F97EB3FF840C6F99B9B15E A3FDFDE0FA OCTET STRING : 133 bytes
228010202	'010300B810000000000000000000000013332313000557365722043657274696600000130B90B00000130E01B32323830313032303200323238303130020100 84A8A0E1FC8F666DBE6CCE351906DC79E2A4144FF8F44B9ABCA8A0C80E6D8A24 2E9A01008185C6D35F46BE6B60EF10D2CA8A2A541A0F9686E8F43EC0C2EFD7BD 231D9B379940 OCTET STRING : 134 bytes
228010203	'010300B810000000000000000000000013332313000557365722043657274696600000130B90B00000130E01B32323830313032303300323238303130020100 8478D310D68DA2B7885EA25289B541BD91A8BDE177E657DB2D29B8A4B50B8C1E FDB1010081C2F3293A51655607778F13207780E166B8028C4220F9DBF9D11B0D 8AB0614A34C0 OCTET STRING : 134 bytes
228010204	'010300B810000000000000000000000013332313000557365722043657274696600000130B90B00000130E01B32323830313032303400323238303130020100 84D14099B66E45BF3FD91D6804D1CAEE280AD7D4D2BD23D0A4DEE16A384D05F3 C2590100818FBBB218A1D2BB96098F124902DC3538294B5E26F38581D7CB57DD ACB43487FDC0 OCTET STRING : 134 bytes
228010205	'010300B810000000000000000000000013332313000557365722043657274696600000130B90B00000130E01B32323830313032303500323238303130020100 84939586B87166A2D6B74483271891BC9EC843CB16196D52743914375ED3ED5A 5A0101007F58911398D86BFDF71E3AB3DF2092E51911282D6207AC03F378E970 5A50DA4114 OCTET STRING : 133 bytes
228010206	'010300B810000000000000000000000013332313000557365722043657274696600000130B90B00000130E01B32323830313032303600323238303130020100

	84C3C461FB53C3CF3B256ED262A7F1FAC04A6119BC660DA3218847108BA2F290 3F40010080AA5AC439B5DAE9199E7E0DC2855BC57C7094995A140E83EA4D4EFA 65BEAFD265 OCTET STRING : 133 bytes
228010207	'010300B810000000000000000000000001333231300055736572204365727469 6600000130B90B00000130E01B3232383031303230370032323830313002010084 84A47C4B7677038FB8283DF81A4602018013DEFC5EF5D26FE80E0787C513F21A AFEF010081895202C9E02A9C775E91C32CE4E5623293A872C4040A64DA34BFAD B4497E4555C0 OCTET STRING : 134 bytes
228010208	'010300B810000000000000000000000001333231300055736572204365727469 6600000130B90B00000130E01B3232383031303230380032323830313002010084 84DC010FA1F115542BB5AA6FEF78F00883B8342CA0152B20B9FCF172577EDC58 1DA801008175B3A0557391ED023821742CE0DD51F5DDBEA56E69F26FFAB51558 65521BC0AE40 OCTET STRING : 134 bytes
228010209	'010300B810000000000000000000000001333231300055736572204365727469 6600000130B90B00000130E01B3232383031303230390032323830313002010084 84C66588002967B509A3803904C6DD81C955BC38CFAE5F85D0C133BEEAAD6696 5EAE010080F51898D3D6433B823F68D6E989C0CDEC8C03E41AB762D3278C7D16 EF2B5FDC54 OCTET STRING : 133 bytes
228010210	'010300B810000000000000000000000001333231300055736572204365727469 6600000130B90B00000130E01B3232383031303231300032323830313002010084 84B321171B105969CFBBA889CF7C287A9BA6ABF2B025431D612B488D89520961 C5B601008199472A7FC3746A43BA45C903DDCA190833D2E213E3F2A6105796F7 CC02D8678500 OCTET STRING : 134 bytes
228010211	'010300B810000000000000000000000001333231300055736572204365727469 6600000130B90B00000130E01B3232383031303231310032323830313002010084 84D6F4F6326D8C3523E90CE5187ED232CE7899B8F9198BF5067A53DC5EBFC6F2 8B6F01008039518904FAEA156247AF9D9D4F55151EE113BEB9B939FB5A76B8ED 8F9C39E3B7 OCTET STRING : 133 bytes
228010212	'010300B810000000000000000000000001333231300055736572204365727469 6600000130B90B00000130E01B3232383031303231320032323830313002010084 84F6C23391F853204DFDEDA69F783C0A7D5CF88428A56EEE0BE78F6065907A4C 518F0100801F16988617BE4D6D0832E3F0A950D1C1C1DAF3B8830C2A33A4D471 0CCEEEFCBB OCTET STRING : 133 bytes
228010213	'010300B810000000000000000000000001333231300055736572204365727469 6600000130B90B00000130E01B3232383031303231330032323830313002010084 840BE004CD22C42C39A4801DB3213CD54DDC43019F352932D0322F4425968FB5 879B0100802F5D17A6A22A3E7E4C7A76F5FF2C77C1D498020DDDFB4A560CAFB7

	0BEE7A19D3 OCTET STRING : 133 bytes
228010214	'010300B810000000000000000000000001333231300055736572204365727469 6600000130B90B00000130E01B32323830313032313400323238303130020100 84C7BFC3F4901ABFE475C7E8D2BBDF5A395FB12C5F5E0DD233EAD339E217F5D4 A98B01008021E349753CEE70EEA59F51D43E439C1AABE4793BF8B93AAB9EDFF6 04EA9FF450 OCTET STRING : 133 bytes
228010215	'010300B810000000000000000000000001333231300055736572204365727469 6600000130B90B00000130E01B32323830313032313500323238303130020100 84A4E5A26394C6EB4BB0B05E5DC0EDBB7F101340248FA95F6483319DD044CF0B A1AE0100812EAB2F890709A8A72222B60D3428858B60FB4617859635E647DF77 CE5CF4E01C00 OCTET STRING : 134 bytes
228010216	'010300B810000000000000000000000001333231300055736572204365727469 6600000130B90B00000130E01B32323830313032313600323238303130020100 84C0D6A39677DDCEC13545232715F54B0B455F3ADA94D1828489AE80E1C47B4B 3C56010081C8F3698BA2852A614A85A5EB65859AA5D5D9A26739C86DE7581220 89B7F2CBF140 OCTET STRING : 134 bytes
228010217	'0102FCB810000000000000000000000001333231300055736572204365727469 6600000130B90B00000130E01B32323830313032313700323238303130020100 825C9403B223847CFA7D1B893FE5E274437B395FBD8939B83352FE3AC8350B1C 71B0100806C93BE27597C5F12D6B9E1E67606D583C71C934A353031F698C34A8 BCDD0CB2C0 OCTET STRING : 133 bytes
228010218	'0102FEB810000000000000000000000001333231300055736572204365727469 6600000130B90B00000130E01B32323830313032313800323238303130020100 83C6CC81E1EF64B22283C89A0C4658FBBEF8E898C9C824DD9EE2C905D93DEFD1 21EC040203E44EC9743BAEADE7B25D73604A2BA629033915A0BC4CE708CD66CF ADF3AC9CC8 OCTET STRING : 133 bytes
228010219	'010300B810000000000000000000000001333231300055736572204365727469 6600000130B90B00000130E01B32323830313032313900323232303130020100 84781354718914B93DA9E2CF214B5A6B9B8E12668582EE0FA56AADF4416B24C5 B411010081873027D04B306844A8A304754527B3BE237E0CA2461E1376577277 87FBF013CE80 OCTET STRING : 134 bytes
222010100	'010300B810000000000000000000000001333231310055736572204365727469 6600000130B90B00000130E01B32323230313031303000323232303130020100 84B58DF7CD47EEC310290DFE3BB693AACC1CF90CBA38C2E328FBAC5CDC4FB50A 7810010081B402C0C0D82566CEDAC55F283712A5589D96A0939686211A1B90D6 8FBB7A15D780 OCTET STRING : 134 bytes

222010101 '010300B810000000000000000000000013332313100557365722043657274696600000130B90B00000130E01B32323230313031303100323232303130020100841A7C952B9C5B100F59CB12AE1966C835CC71D8D36801A07D244671A06DDD78782E01008198F276520C6E3C617B8C900809882B042FF6C1AC82585779D3153F8EEDEEF22D40
OCTET STRING : 134 bytes

222010102 '010300B810000000000000000000000013332313100557365722043657274696600000130B90B00000130E01B32323230313031303200323232303130020100884020A96EFD3C3427F70B58ECA428030B24DF2E2406FB944B7B72325EDD598BA7B85010080C9C1FD3758B9F59DF5A89E0CE706957775374F8621892AF164011D47F93B0DDC
OCTET STRING : 133 bytes

222010103 '0102FEB810000000000000000000000013332313100557365722043657274696600000130B90B00000130E01B32323230313031303300323232303130020100883849DE1F5B33763CCBA2A1E52CA64FB594EC904F2DF66C62E32D5F569589C22615804020622C9BFE8AA2CA7068D2994D0E24865BE4E93555C94CF88830A6DB01546DA7669
OCTET STRING : 133 bytes

222010104 '010300B810000000000000000000000013332313100557365722043657274696600000130B90B00000130E01B32323230313031303400323232303130020100884954FBA171F4154D5A5A23C76631A1AF9D1B1787527BE0C403A8AA22B5D7A10D92A0100814A4F130176625539E05AD8A5E033C596E29806841B2E08275707F016C673651880
OCTET STRING : 134 bytes

222010105 '0102FEB810000000000000000000000013332313100557365722043657274696600000130B90B00000130E01B32323230313031303500323232303130020100883DFF81B8469FE12A07C9D05B42712F37E0DFF90470688577C697C34665FB7E9102C040205F7F53A56FED16CC59E759D79222F9BA3972B17E386AB056879007660DCDBB242
OCTET STRING : 133 bytes

222010106 '0102FEB810000000000000000000000013332313100557365722043657274696600000130B90B00000130E01B32323230313031303600323232303130020100883C5B5B5517B598534955FF9185CA5D07AB86C88FF8D68D9670A3D7D40A2E0EB1F8C04020251F95E6ACC141723C8D1657AE4C32E90D97846C96041F3CB13E36CEFE86456CC
OCTET STRING : 133 bytes

222010107 '010300B810000000000000000000000013332313100557365722043657274696600000130B90B00000130E01B32323230313031303700323232303130020100884B2AE38E3EB571C0C7167AD958071283FE05B00B1D891CE511814535E8799B2EA5B0100819F4C7881182A5E8AEED5D41999D25B1C935030EC4EF44BCC605C3AE739D57DEBC0
OCTET STRING : 134 bytes

222010108 '010300B810000000000000000000000013332313100557365722043657274696600000130B90B00000130E01B32323230313031303800323232303130020100

	848C9127E5E5B561A1441A52679099C5162F98DEC8D7716FB9D0328D89C26D86 B9A201008197D8B9D882B0C55BF6DD84A2B4E059D68DC032DE43629CBFD18E06 5E7061B47A80 OCTET STRING : 134 bytes
222010109	'0102FEB810000000000000000000000013332313100557365722043657274696 6600000130B90B00000130E01B32323230313031303900007 56E646566020100 83A78BCE53BD6F9B3A84C899A2DE6F13DDAC40941177B325F9637CDFEDC003F4 EB40040202334E31DD5C681BB1B14365E00667D7C2A32B5E7AA115C0F394EFAE CA848FC810 OCTET STRING : 133 bytes

A4.3.tmuis, ksu

Note that the tmuis consists of the concatenation of serviceProviderId and userIdSP

IMUI	serviceProviderId (5 bytes)	userIdSP (4 bytes)	ksu (16 bytes)
228010200	'3232383031	'90E42BC3	'F32B31A0F51BC1E759DAE06A861C32DC
228010201	'3232383031	'0EE37537	'0CAA399E372E3991239CD984305ACF64
228010202	'3232383031	'D39B2111	'1C6E0EE55428877C15F398ED9B6CAC68
228010203	'3232383031	'EFC67CFA	'FD3879094EE964BF65D442C0C206D88A
228010204	'3232383031	'D8621245	'8F5A0C2F8E7B86873848E2D70B2E67A2
228010205	'3232383031	'F4DE2C23	'D85AED0996B26BFEBD8481985E912551
228010206	'3232383031	'884E4547	'3724B3F74C1444C2F45D96DCC3FB6510
228010207	'3232383031	'F26A988E	'93562AF4CAD66ADCC7DAA2C77F216DB9
228010208	'3232383031	'E3E606C2	'4EDFBD239B32898ACA52BBF158E60951
228010209	'3232383031	'F505A881	'9E443EAE57335ED0C2725A99DF0E9543
228010210	'3232383031	'54036CC6	'B76F7C7368A7F2BC72F4F710EE4A7E1A
228010211	'3232383031	'2857C5C5	'0DCDFDC9BD08B7A836B168D0959651D1
228010212	'3232383031	'7DAA3D89	'DE602DA6ACC9310E86781593EB0201B5
228010213	'3232383031	'A099B688	'2E7465428DD2CF3248CB59D3666686F3
228010214	'3232383031	'EC5A6CEB	'BDD87D2268A052FA6D54A99097620CA9
228010215	'3232383031	'601B995A	'2AF9CD628D67B047F4F96C34C67B2A6D
228010216	'3232383031	'9D46196A	'6E61E303DB0BF160BF5ECCFDF9D78AAB
228010217	'3232383031	'F8E42832	'BCB204A6E275A195EA20EDC3601B5C71
228010218	'3232383031	'AD026DCB	'E15232442AB4DBC1C2AA2FC45422D70E
228010219	'3232383031	'315EB9D8	'42C7582603C348956B368EFFD15ECCA2
222010100	'3232323031	'E2D04097	'1F6CE9B1D70C16CA80CAC3577FC28D54
222010101	'3232323031	'4062FCFF	'312F4C954AC1C8E5ED3A27F6482752E3

222010102	'3232323031	'721A8AB3	'46FD6C1180580F4EC255A2E2B9A5244D
222010103	'3232323031	'82DF611D	'702A07BD719196EF990998845E9BF3D4
222010104	'3232323031	'73F233F6	'2B3E42D5BB651CF5C94F34F3DFEF9FF9
222010105	'3232323031	'58B95D78	'B6F8DC878B969592E13E25E11A3919D4
222010106	'3232323031	'A7ABD61B	'52AFDB9FB94B521E7C73544A88DCE96B
222010107	'3232323031	'D75888E3	'F206DC7A9ED0B8616BA0E6620AFF79C9
222010108	'3232323031	'E9A9CB35	'A49482C42A89E48FCB4ED799040C1D03
222010109	'3232323031	'A42491F5	'ACA67C7CC1F41EA29D3CDAF4DEDC2CB8