IST-2000-25350 - SHAMAN

Deliverable Number	D13 – WP4 contribution
Deliverable Title	Final Technical Report – Secure Module
Date of delivery	30-NOV-2002
Document Reference	SHA/DOC/GD/WP4/D13A4/1.0
Contractual Delivery Date	30-NOV-2002
Editor	Hans-J¸rgen Heinrich (Giesecke & Devrient)
Participant(s):	ATEA, Ericsson, Nokia, RHUL, Siemens AG, T-Nova, Vodafone
Workpackage	WP4
Est. person months	
Security	public
Nature	report
Version	1.0
Total number of pages	46

Abstract:

This document describes a concept for a security module which provides high security for future mobile communications in heterogeneous networks. An overview of state-of-the-art technology and an analysis of the functionality requirements for secure network access and personal area network internal communication are given. From these requirements, a reference model of a security module introducing several security levels is developed. The functionality split between security module and mobile equipment for selected application scenarios is analysed in detail.

Keyword list:

Security module, smart card, security, functionality split, tamper-resistant device, personal PKI

The work described in this report has been supported by the European Commission through the IST Programme under Contract IST-2000-25350.

Executive Summary

The purpose of workpackage 4 is the development of a concept for a security module which provides high security for future mobile communications in heterogeneous networks.

A security module is a tamper resistant device that is both physically and logically secure and has the ability to contain data and/or perform functions for certain security systems. The use of security modules for holding sensitive data is already widespread; the most obvious example for an application making use of security modules is GSM.

The downside of smart cards, as well as of all tamper-resistant devices, is the shortage of storage and processing power. However, for the following 5 years, a strong improvement of performance can be expected; the chip's clock rate will probably rise from typically 1-5 MHz today to 20-33 MHz, the storage capacity of the EEPROM from 16 kB to 32-96 kB and the transmission rate which is 9,6 kB/s across a serial interface will presumably reach 115 kB/s.

A security module must meet several requirements in the SHAMAN context. For WP1, the security module must support secure network access. The functionality split for one secret key (AAA for IPv6) and two public key protocols (IKE and JFK) was studied. In addition, the involvement of the security module in security mechanisms for protecting the confidentiality and integrity of the access link was investigated. The security module is not directly involved in the protection of the access link communications.

For WP2, work focussed on secure private area network (PAN) internal communication and distributed security modules. It was studied how the security module is involved in the imprinting process. Additionally, it needs an authentication algorithm for connection establishment, whereas confidentiality and integrity protection on it are not practical. The workload of security functions on a security module can be distributed either on a group of modules or between a single module and an untrusted device. The distribution of a single private key operation between the security module and an untrusted device was described. Delegated authorisation of the RSA private key was also studied.

For WP3, it was investigated how a security module can implement or support a personal CA (certification authority).

With today's technology, not all of these can be implemented on a single device. It is necessary to find a trade-off between high security on the one hand and feasibility as well as performance on the other hand. Therefore, 3 different levels of security have been defined which form the basis of a security module reference model. The levels are called “intermediate”, “high security” and “personal-CA level”. For the intermediate level, functions like random number generation, storage of long-term secrets and calculation of a one-way function are required. In addition, basic protocol support for a secret and a public key protocol must be present. For high security, it is also demanded that the SM be able to store short-term secrets, compute Diffie-Hellman secrets, create public-private key pairs on card and store security contexts (for plastic roaming). Besides, some more advanced (and more secure) protocol options must be supported. For personal CAs, the smart card should also be able to manage certificates, i.e. store, create and validate them, including revocation checks.

Today's smart cards already offer an appropriate basis for the realisation of a SHAMAN-compliant security module. All of the intermediate level and some of the high level functions can be provided. The rapid increase of storage capacity and processing power will allow more sophisticated solutions in the near future, offering the user a personalised token with high flexibility and maximum security.

Table of Contents

1 Overview

The purpose of workpackage 4 is the development of a concept for a security module (SM) which provides high security for future mobile communications in heterogeneous networks.

Previous SHAMAN deliverables have developed the SM concept: D05 - *Intermediate report on the role of Security Modules in heterogeneous networks distributed terminals and PKI* [1] and D08 - *Intermediate specification of Security Modules* [2].

A security module (SM) is a flexible and secure unit that will provide trustworthy storage and processing of sensitive or critical data in future mobile communications. It consists of both hardware and software components which together provide a trusted device. The characteristic feature of the hardware is that it should be highly tamper-resistant; the software should be able to maintain its trust status by controlling the downloading of further code both into itself and into dependent components of its environment which rely on it to maintain system assurance.

The SM must be designed to be easily administered and portable. The technology will develop from existing smart card engineering by providing enhanced processing, storage and input/output. The internal architecture will be such as to support measurable and certifiable assurance and trust.

As there will always be constraints on SM performance compared with surrounding components, techniques for subcontracting tasks and for splitting functionality will be required.

In the following, a functional definition and model of the SM will be developed and services to be provided will be identified.

Chapter 2 gives a definition of the term “security module” and describes its principal functionality.

Chapter 3 is devoted to the technology trends and use of smart cards in telecommunication as we see it today.

In chapter 4, the requirements for security module which arise from the SHAMAN project are listed.

Chapter 5 describes in more detail the functional requirements for a SHAMAN security module as identified in workpackages 1 – 3.

Chapter 6 gives a reference model for the SHAMAN security module which forms a basis for implementation.

The functionality split for selected protocols is described in chapter 7.

Chapter 8 gives a conclusion and outlook.

2 Definition and principal functionality

Security modules are a flexible and secure mechanism to allow for secure and personalised mobile communication within today's mobile communication infrastructure. They must be designed to be easily administered and portable.

In the following, a definition of the term "security module" will be given and the services they provide today will be briefly identified.

2.1 Definition of the term "security module"

A security module (SM) is a tamper resistant device that is both physically and logically secure and has the ability to contain data and/or perform functions for certain security systems. A security module is capable of storing secret data and executing security functions in such a manner that no information about the secret data that could be efficiently used to break the security system is leaked out from the security module. For example, smart cards are regarded as suitable devices to be used as an SM. It is not necessary that the module must perform all functions that use secret data on its own. In fact, in some cases execution of security functions can be distributed between the SM and some other devices without revealing information about the secret data outside the SM. The module can be a single smart card, but does not have to be. It can also consist of multiple smart cards or a smart card which works in cooperation with an external server. To be tamper-proof, the communications between multiple smart cards or between a smart card and an external server must be trustworthy. All this means that the security module does not have to be a single physical entity: It can be distributed across several locations or at least be accessible from several locations. Therefore, secure communication between the locations must be granted, like in a PAN (Personal Area Network, see Deliverable 3 [3], chapter 2).

2.2 Current use of security modules

The use of security modules for holding sensitive data is already widespread in a variety of applications supporting a number of services. Security modules have evolved and are now capable of doing more than just holding data securely. Many security modules are depended upon to do cryptographic computations, generate encryption keys and carry out authentication, as well as to just hold data.

Having become an integral part of the GSM world, the security module is required to authenticate itself to the network on the behalf of the owner of that module before access to network services can be granted. The security module is also used to generate and store encryption keys, which are used to protect user traffic and signalling data on the radio interface.

In the financial sector, banks are now issuing cards to their customers which contain a security chip for enhanced security as well as the ubiquitous magnetic strip. There are many other applications of security modules, e.g. for access control to buildings and computers (single sign on).

2.3 Services to the terminal

Security modules are defined to be tamper resistant devices that can perform data processing and store confidential information. A terminal (PDA, Mobile handset) is generally not as physically or logically secure as a dedicated security module when it comes to performing specific security operations. It is typically easier to extract data from a terminal than from a security module. However, on the other hand a terminal can hold more data, is more powerful and is capable of supporting a larger range of security functionality.

It is therefore often necessary to split security functions between the security module and the terminal to ensure an adequately secure yet efficient implement. In this case the security module has to support some security functionality and the terminal may use the result of this computation. An example of such integration is the WAP Identity Module (WIM) [4] defined by the WAP forum. The WIM has support for processing and storing security data, it is able to do digital signing and will soon support encryption where the signed (and encrypted) data is then passed to the terminal and additional security mechanisms can be applied (e.g. transport layer encryption using WTLS [5]). For example, the WIM is used for signing data with a user's private key which is stored in the WIM, the terminal carries out the verification of digital signatures where the root certificates can be stored on the terminal or in the WIM.

Another example of splitting functionality between the SM and the terminal is the GSM encryption method. The mobile station is required to use the session key generated by the smart card based subscriber identity module (SIM) to encrypt user traffic (e.g. voice and data). Figure 1 shows that the session encryption key (Kc) is generated in the SIM by inputting the long-term subscriber-specific authentication key (Ki) and a random (RAND) that has been received from the mobile switching centre (MSC) into the A8 algorithm. The session key is then passed to the mobile station (MS). The mobile station uses this session key to encrypt the voice data using the A5 encryption algorithm, which the MS must support. The session key is input to the A5 encryption algorithm along with the plaintext voice data to produce encrypted user data. The long-term authentication key Ki never leaves the SIM, but is used to generated a session key Kc.

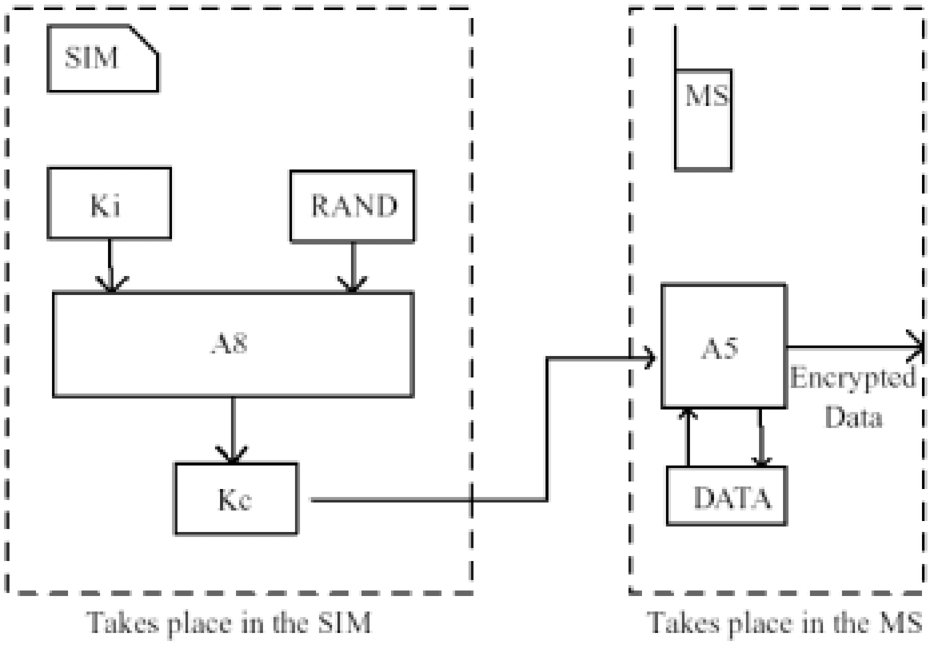


Figure 1: Functionality split between SIM and MS for GSM encryption

2.4 Services to the network

Providing services to networks is the most common use for security modules today. The wireless world, in particular GSM, relies on a security module, known as the Subscriber Identity Module (SIM), for authentication to the network based on a long-term subscriber authentication key before a user is allowed to use any service from that network. The security module is required to encrypt the voice call with the symmetric key generated in the SIM. The SIM therefore is an integral part of the GSM architecture, providing security data storage and cryptographic processing. The Universal Subscriber Identity Module (USIM) is the 3G version of the SIM to work with the Universal Mobile Telecommunications System (UMTS). It supports 3G protocols and is backward compatible to support 2G authentication methods for access to 2G networks +[3]. The USIM is not regarded as a physical entity, it is seen as a logical application that resides on a Universal Integrated Circuit Card (UICC). The UICC contains one or more USIMs and possibly other applications (e.g. credit card functionality or WIM). By inserting the USIM-card into a UMTS terminal the user is recognised by the UMTS network and can be addressed on this terminal via his personal telephone number (MSISDN).

There will be a multitude of different types of terminals in UMTS, e.g. multi-mode or multi-band handsets, notebook-like communicators or UMTS-laptops with camera, speakers and microphone all equipped with a USIM-card. There will be terminals too where more than one UICC can be inserted.

LAN services have also adopted the use of security modules for secure log on. The security module like in the GSM world authenticates the user to the network in order for the user to use the services offered by the network. The security module can also be used to sign e-mails and hold encryption keys to encrypt e-mails and other data. An example of this is the Windows for smart cards platform [7], it allows smart card authentication and encryption with the current windows operating systems. Windows for smart cards can be programmed for one or more users, but a user cannot have concurrent account usage. The smart card is used to authenticate a user to the PC or a network. The security credential such as a password or a biometric value is stored on the smart card this is compared to the template stored on the network as with an ordinary secure log on system.

2.5 Services to the end user

The use of security modules for the end user is useful when private and confidential data needs to be stored. Presently security modules are used in various ways to hold different types of information for a user. Standards such as PKCS#15 [8] allow for storage and access to data in a secure manner.

The WAP Identity Module (WIM) (see Figure 2) is used in performing WTLS and application level security functions. It stores and processes information needed for user identification and authentication. Sensitive information such as private keys are stored on the WIM and all operations which involve these keys can take place on the WIM. An example use case of the WIM can be in generating a WTLS session key where the WIM generates a random, takes the random generated from the server, applies a function and outputs a session key for encryption of the WTLS session.

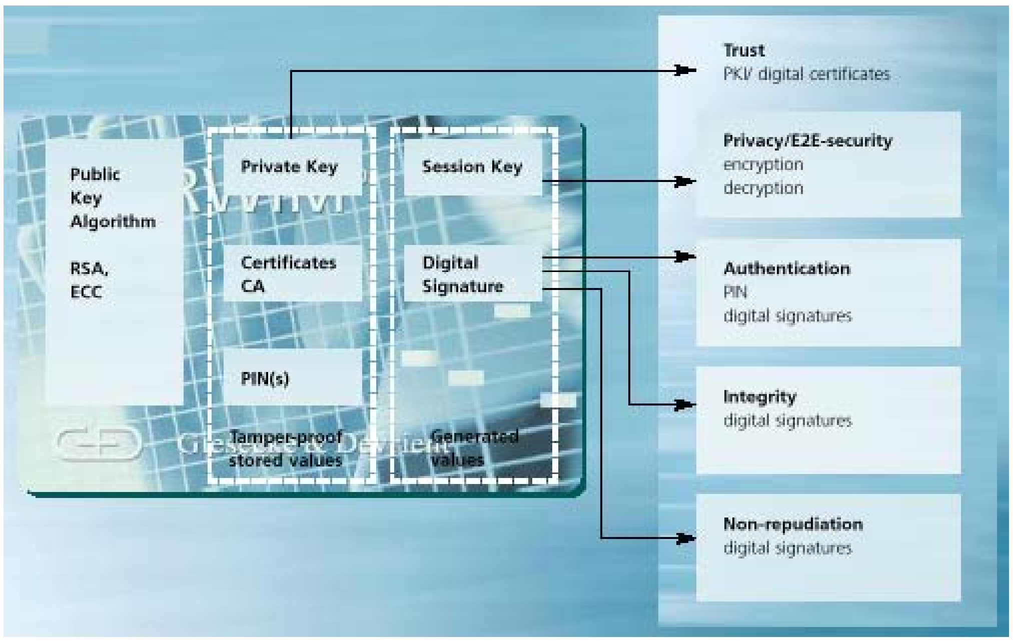


Figure 2: Structure of a WIM

Other functions include SignText, which allows a user to sign any piece of text. In this case a hash of the text to be signed is passed to the WIM from the terminal, the WIM then uses the stored private key to apply the user's signature to the hash, and this is then passed back down to the terminal; the private key never leaves the WIM. The WIM follows the PKCS#15 [8] file structure in order for it to carry out its requirements to store the following data:

- Information on properties of the module: supported algorithms etc.
- Key pairs for authentication, key establishment and digital signatures

- Own certificates for each key pair
- Trusted CA certificates
- Data related to WTLS sessions (including master secrets)
- Information on the protection of data with PINs

Cryptographic Token Information Format (PKCS#15) specifies how keys, certificates and application specific data may be stored on an ISO/IEC 7816 compliant IC card or other media. It provides a hierarchical directory structure for the storage of information with certain directories having access control rights put upon them. PKCS#15 allows for a security module to present security credentials such as a digital certificate to an end application in a standardised way. PKCS #15 has the following distinct characteristics:

- Allows multiple applications to reside on the card
- Supports storage of any type of objects (keys, certificates and data)
- Support for multiple PINs whenever the security module supports it.

PKCS#15 defines a directory structure that allows for the storage of particular security information. It does this by defining directories such as PrKDF (Private key directory file), PuKDF (Public key directory file) etc. These directories can be accessed in a standardised way by PKCS#15 compliant hardware. An important file in the structure is the Authentication Object Directory File (AODF), this is the directory where the authentication objects are kept (e.g. PINs). The WIM specification specifies that at least one AODF must be present on a WIM. The first object in the AODF is considered to be a general PIN (PIN-G), if not otherwise indicated, all relevant files (CDF, PrKDF) are protected with this PIN.

2.6 References

[1] ‘Intermediate report on the role of security modules in heterogeneous networks, distributed terminals and PKI’, Deliverable D05, IST-2000-25350-SHAMAN

[2] ‘Intermediate specification of security modules’, Deliverable D08, IST-2000-25350-SHAMAN

[3] ‘Interim Report - Security Architecture for Future Mobile Terminals and Applications‘, Deliverable D03, IST-2000-25350-SHAMAN

[4] WIM Specification, www.wapforum.org

[5] WTLS Specification, www.wapforum.org

[6] 3GPP TR 31.900, www.3gpp.org

[7] Windows for smart cards, www.microsoft.com/windowsce/smartcard/start/intro.asp

[8] PKCS15, http://www.rsasecurity.com/rsalabs/pkcs/pkcs-15/index.html

3 Technology trends and use of smart cards in telecommunication

In this chapter we examine different kinds of security modules related to mobile telecommunications. Security modules that are taken into consideration in this chapter are smart card (e.g. SIM, WIM, USIM), iButton, eToken and secure ID. This chapter also contains a description of work that has been done by IETF.

Smart card solutions have a very essential role in mobile telecommunications. There exist also several ongoing EU IST projects on smart cards [9]. The hard- and software trends in smart card technology are reviewed in this chapter.

3.1 Tamper resistance of security modules

One of the essential concepts and concerns in security device technology is tamper resistance. The general definition is that devices as well as their data and algorithmic behaviour can not be manipulated by, copied by or revealed to any unauthorised person. Tamper attacks can originate from third parties or from inside the system. The attack can be started at any one of the following access levels of a device:

- physical access to device with damaging analysis tools
- physical access to device with non-damaging analysis tools
- direct access to device during operation
- passive access to device during operation
- direct or passive access to external operations or protocols of a module

Basic tamper attacks to SM devices with damaging analysis tools are for example done by physically attacking the device. This can happen by unwrapping any packaging or analysing the physical structure of the device by hardware manipulation, by tapping signal lines, by contacting areas on a chip, by analysing electromagnetic radiation or by willingly inducing effects or errors (by light, laser, radiation, ...) into the operating module. Results or any different behaviour after manipulation can give useful information for analysis of the entire device. There exist examples [10] of incidents with devices analysed and manipulated by this approach. Most of these successful attacks are on offline systems like pay-tv descrambler or similar systems. Up-to-date security module technology provide the means to successfully prevent such attacks.

Older attacks based on timing analysis aim to detect timing differences for selected operations like checking wrong or correct PINs. These attacks have already been know from early mainframe architectures and are successfully solved now. In recent years a new class of attacks has showed up: Differential Power Analysis (DPA) and Simple Power Analysis (SPA). Both SPA and DPA try to analyse data and behaviour of algorithms by detailed measurement of the power consumption of devices in use. They are based on the finding that different operations, different data sets and different algorithmic behaviour causes differences in the power consumption of a microprocessor executing any code. In measuring the detailed power consumption during operation one can easily detect loops and cycles as well as different kinds of memory access and differences if zero or non-zero values are written or read. A close differential analysis of the power consumption can even reveal secret keys of a device if no guards against such attacks are implemented in the hardware or software of the security module.

Today’s security module technology aims to disallow the analysis of the modules to prevent the production of visually, functionally and operationally identical copies of the device. While some of these feature are achieved by secure packaging including manipulation detectors, similar techniques are applied at the logical level using secure operating systems on smart cards, or by designing algorithms to make SPA/DPA impossible.

Secured operating system can for example repeatedly check for internal consistency and authenticate with and against external devices during any external communication, to detect potential manipulations, while carefully designed algorithms disallow SPA/DPA and hardware sensors prevent physical manipulation of the device.

Beside the high efforts needed for producing and implementing tamper resistant security devices, any certification of a security device for tamper resistance is a time consuming and expensive process as well. Limiting the certification process to a smaller device allows for an easier and less complex verification process and for reuse of the security module within multiple devices later on.

Additionally it also allows for easy adoption to local legal requirements, which is essential for signature or encryption functionality in most countries. Using a removable and exchangeable device allows the reuse of certified technology within new or updated but not yet evaluated environments or devices. Additionally, removable tamper resistant devices allow for secured transfer of secured information within possibly insecure environments by simply transferring the plug in module.

3.2 Smart Cards

There are two general categories of smart cards: contact and contact less. A contact smart card requires insertion into a smart card reader with a direct connection to a conductive micromodule on the surface of the card (typically gold plated). The chips used in all of these cards fall into two categories as well: microprocessor chips and memory chips. A memory chip can be viewed as small floppy disks with optional security. In mobile telecommunications contact smart cards with microprocessors are used.

GSM is the major application of smart cards. Instead of a mobile phone being personalised to a subscriber, as in first generation analogue systems, the handset is generic to all subscribers and is personalised by means of a Smart Card SIM (Subscriber Identity Module). The SIM contains all the personalisation and encryption details for the specific end user.

The SIM contains a microprocessor capable of handling its own security and managing the flow of data within the chip and between the chip and the outside world. It requires an operating system to manage these functions. This operating system must conform to international standards set by the SMG9 group of ETSI.

In UMTS subscribers are also personalised by a smart card like in GSM. This smart card is the USIM (Universal Subscriber Identity Module). USIM contains new security algorithms; for example, it allows mutual authentication of subscriber and serving network.

The Wireless application identity module (WIM) is used to provide security on the application layer in WAP. WIM implementation can be any tamper resistant device, not necessary a smart card. Most likely the WIM is integrated on a SIM or USIM.

3.3 Smart card hardware

3.3.1 General

Telecom security modules for mobile equipment are currently based on SIM cards containing standard low cost and low power microprocessors equipped with ROM, RAM, EEPROM storage and hard coded security algorithms. The security algorithms are based on symmetric key cryptography with individual permanent secret keys securely stored on both the SIM cards contained in the mobile device and the operator’s centralised databases.

While the operator’s centralised data processing centres can easily be protected and secured by standard means, the secret keys stored in the mobile device need to be protected by a tamper resistant SIM card to protect against an attacker obtaining the authentication key and using it to masquerade as the legitimate user.

SIM cards are tamper-resistant and trusted devices, produced by certified and trusted companies following agreed standards and state-of-the-art technology. SIM cards act as a proxy for an individual user and can be easily transferred to other devices by the user. Beside this the SIM card is the property of the operator and all security features and services are set-up, guaranteed and certified by the individual operators.

The hardware technology of smart cards as a low cost, embedded consumer product can be estimated to follow the hardware technology of high-end standard off-the-shelf microprocessor technology with a delay of approximately 8 years. Production technology is at least 2 generations behind current state-of-the-art in semiconductor production. First generation high-end productions are used for memory components, second generation fabrications are in use for standard microprocessor products, while third and forth generation fabrications are used for telecom and embedded devices. This approximate delay of 8 years gives a clear outlook of what can be expected from core technology during the next few years. Additionally, specialised and standardised smart card components are introduced with a lower delay than 8 years. This effect can currently be seen with additional interfaces (e.g. USB), crypto-coprocessors (e.g. for RSA [11] or PKI) and soon to be released Java byte code interpreters.

3.3.2 Processing power

High end devices for smart card products are currently projected for 33 MHz CPU operation in 2002 by various chip suppliers and will reach 100Mhz by 2005. Restrictions from the mobile equipment and by the telecommunications standards limit the clock rate to 8 MHz or selected multiples to reduce noise on critical frequencies. Especially independent processor clocks adapting the CPU clock rate to the individual processing power needed during peak times and allowing slower low power operation during off-peak times are not permitted today but this technology is already available in today’s chips. Fixed multiples of selected frequencies are allowed, but additional restrictions arise from heat dissipation and power consumption of individual devices. Currently 8 bit processor technology is being replaced with 16-bit RISC technology and the transition to 32-bit technology can be foreseen within the next 2 years.

3.3.3 Storage

Today’s standard devices are still using 32kB ROM, 2kB RAM and 32kB EEPROM. While low cost products are still using smaller amounts of memory, there are high end smart card products available providing 128kB of ROM, 4kB RAM and 64kB EEPROM. Any scale in memory size directly scales to the chip area needed for the device and therefore scales to the price of the product. This especially holds for high volume production periods like in year 2000, but is an major impact on prices in current low production volume period too.

The roadmaps of various chip suppliers show that new products during the next 2 years mainly increase the size of RAM (up to 10KB), while retaining the ROM and EEPROM sizes at today’s high end sizes (128kB ROM, 64KB EEPROM). As a new feature the FLASH ROM technology will be introduced in 2002 with additional 192kB of memory available in FLASH technology during the next 2 years. Besides the additional memory space available, the devices can now be programmed and used more individually. Updates, enhancements or replacement of operating systems and services can be completely done after initial chip production without the need of an additional ROM mask redesign and chip production cycle. But a complete update of operating systems during smart card lifetime can not be expected during the next few years for security reasons as well as for technical reasons.

Additionally memory management and protection units (MMUs) are currently introduced to the consumer market allowing for hardware based protection and mapping of individual memory areas to the applications and operating systems needs. Up to now these protections are assured by a secured operating system implementing access restrictions by software means, which is technically feasible, but causes a large overhead in verification of the mechanisms. By using MMUs the operating system can now make use of individual hardware protection mechanisms to protect certain memory, code and interface areas from program access at the operating system or applications program level.

In long term (5 years) FERAM technology will replace the EEPROM technology and provided larger memory sizes with characteristics only known from today's RAM technology. This includes memory access cycles of 1 micro second compared to today's EEPROM access cycle time of 1-3 micro seconds.

3.3.4 Interfaces and bandwidth

While there are several attempts to provide faster interfaces for other applications of smart cards (e.g. USB Interface for signature or computer logon cards), the standardisation of mobile equipment restricts bandwidth to the available serial transmission rates on the existing contact pads of GSM smart cards. The serial line of state-of-the-art smart card processors can be programmed to provide up to 115.2 kbit/sec bandwidth and as an extension one existing, but commonly unused, pad on the external interface can be re-used as a second serial I/O-channel. Hereby the I/O bandwidth can be significantly increased for special usage without violating the standard interface definitions for the regular I/O line.

Of additional importance is the memory bandwidth if larger amount of data (e.g. key sets) have to be stored in secured memory areas. Currently the memory bandwidth is low compared to existing DRAM technology and no significant improvement can be expected for the very near future. Storing single byte or single page data currently takes some milliseconds with today's EEPROM technology. In fact even the existing processing power of the most recent smart card CPUs cannot be fully used, since both the external serial interface and the internal memory interface are slow compared to today's demand from data streaming format. For example, full data stream encryption is only possible for slower streams using today's technology. The external I/O bottleneck will be removed soon by USB like interfaces, but GSM specifications do not yet allow this. The memory bandwidth will be significantly increased by FERAM technology in the longer run (5 years), allowing smart cards to make use of their already existing processing power.

3.3.5 Architectural support for software

Architectural support for software is provided by various means. The already mentioned memory management units as well as basic segmentation concepts support the implementation of access checks in operating systems software. Another important area is the provision of architectural support of cryptographic algorithms by implementing cryptographic co-processors for DES and consequently for 3DES. Recently RSA crypto engines have been introduced, which currently support 1024-bit key length and will support 2048 bit RSA keys in 2002. More elaborate and advanced crypto processors can be expected within the next 5 years as plain commodity products.

A more high level support of software concepts is the provision of so-called HW accelerators for selected operating systems or languages. The roadmaps of chip suppliers show HW accelerators for Java (HW byte code interpreter) to speed up Java code execution or to ease Java virtual machine integration by providing a large set of the basic virtual machine operations in hardware. Even operating systems like MULTOS or WINSC could gain from the provision of individually provided HW accelerators. It is not yet finally decided if Java is the open operating system platform for long term use on smart cards. However, it adapts well as an implementation, programming and interfacing language for scalable embedded devices for the next few years. In the long run even open source projects like LINUX could influence the smart card industry. It depends on the development of an embedded LINUX version and the resolution of stability and security issues.

3.4 Smart card software

The current state-of-the-art in software technology and operating systems for smart cards is very similar to UNIX consolidation 10 years ago where some different implementations and many different branded products merged into a consolidated UNIX release, Windows NT and LINUX.

While vendor-specific OS still dominate the smart card market, so-called open operating systems have been discussed for some years and the first trials have been successfully started in the past. Currently, the important open platforms are MultOs, Windows for Smart cards and Java for Smart cards. While Java has already proved to be a viable concept for the future, Windows and Java platforms are still under heavy development and MultOs development has settled down. For open platform concepts the importance of standardised APIs is recognised as state-of-the-art and standardised APIs replace the individual vendor-specific APIs or clean up the existence of multiple API variants on smart card platforms.

Today’s commercial systems are based on symmetric key algorithms where the secret key stored on the SM is shared with some other entity. In this case security can be compromised without attacking the card if the secret key can be obtained from the other entity. More recent systems start to make use of asymmetric key algorithms where the private key stored on the SM does not have to be revealed to any other entity. In this case both entities must have their own private key and must be able to obtain an authenticate copy of the other party’s public key. When large numbers of entities want to communicate with each other securely this often leads to the development of a so-called public key infrastructures. In a PKI all recipient public keys needed to send encrypted messages or verify signatures are stored as public key certificates in a global public directory with various lookup mechanisms. The general assumption behind this is that knowledge of a public key does not allow the corresponding private key to be derived.

PKI requires increased computational power from SIM cards as they have to store a larger amount of secret data on the card and handle more dynamic cryptographic data interfacing with the crypto unit. Therefore PKI puts a load on storage, interface and computational power. Additionally PKI allows for “non-repudiation”, which allows third parties to verify that only a certain person or entity could have performed a specific action. This can be done using digital signatures, since only the entity that knows the private signing key can generate signatures. This feature also has consequences for the smart card production process so that keys and PINs are only stored on the card and nowhere else. With the symmetric cryptography approach the key has to be stored outside the card, for example, so that it can be transferred to the network operator for loading on the authentication database in the case of GSM security. In contrast to the symmetric cryptography systems, no second copy of a secret key should exist in a PKI system, thus nobody can misuse any copy of a secret key. But it should be noted that PKI systems rely on the security of the underlying cryptographic algorithms to ensure that the private key cannot be derived from the corresponding public key or from any other public information such as signatures generated using the private key.

3.5 Expected scenarios

With regard to pure single chip performance the following figures (see Table 1, Table 2 and Table 3) are expected in the next few years for lowest cost, mass-market and sophisticated application (and price) scenarios.

	lowest cost	mass market	sophisticated
Technology	0.8 μ	0.5 μ	0.25 μ
MHz	1	1 -5	10
RAM	0.25 kB	1 kB	2.5 kB
ROM	16 kB	32 kB	96 kB
EEPROM	8 kB	16 kB	64 kB
FERAM	None	None	None
Coprocessors	None	None	RSA, DES
External IF	Serial 9.6 kb	Serial 9.6 kb	Up to 115 kb
Storage cycle time	8ms	4ms	2ms

Table 1: Current scenario

	lowest cost	mass market	sophisticated
Technology	0.35 µ	0.25 µ	0.18 µ
MHz	1 - 5	12	66
RAM	0.25 – 0.5kB	3 kB	12 kB
ROM/ Flash EEPROM	16 kB	128 kB	256 kB
EEPROM	8 kB	32 - 64 kB	128 kB
FERAM	None	None	None
Coprocessors	DES, AES	DES, AES, RSA, ECC	DES, AES, RSA, ECC
External IF	Up to 115 kb	Up to 115 kb	Up to 115 kb/ USB
Storage cycle time	3ms	2ms	2ms

Table 2: Scenario expected for next 2 years

	lowest cost	mass market	sophisticated
Technology	0.25 µ	0.18 µ	0.12 µ
MHz	1 - 10	20 - 33	120
RAM	0.5 kB	6 kB	(64 kB)
ROM/ Flash EEPROM	24 kB	256 kB	(512 kB)
EEPROM	8 - 16 kB	32 - 96 kB	(256 kB)
FERAM	None	None	1 MB*
Coprocessors	AES, ECC	AES, RSA, ECC, Hash	AES, RSA, ECC, Hash
External IF	Up to 115 kb	Up to 115 kb/ USB	Up to 115 kb/ USB
Storage cycle time	2ms	1ms	(1ms) – 100ns

Table 3: Scenario expected for next 5 years

3.6 Other configurations and formats

3.6.1 iButton

The iButton is a 16mm computer chip armoured in a stainless steel can. Just like smart cards there are memory iButtons and iButtons with a microprocessor. Java powered cryptographic iButtons contain a microprocessor and include a complete cryptographic module in a Java Virtual Machine (VM) that is Java Card 2.0 compliant. JavaCard 2.0 has been loaded onto an iButton, which has been mounted on a ring. Much like a Smart Card, the Java-powered ring uses a unique digital signature to encode its owner's identification. Unlike the Smart Card, the ring is durable, wearable, and supports more memory, replacing at least 10 cards in its ability to initiate multiple transactions.

3.6.2 SecureID

RSA SecurID hardware tokens provide tamper resistant two-factor authentication, based on RSA Security's patented `Time Synchronization` technology. This authentication device generates a simple one-time authentication code that changes every 60 seconds. To access protected resources, the user simply combines his secret PIN with the code generated by his token. The result is a unique, one-time code that is used to positively identify, or authenticate, the user. If the RSA ACE/Server validates the code, the user is granted access to the protected resource. If the code is not valid, the user is denied access.

3.6.3 Credential carriers

Credentials, as defined by the IETF SACRED group, are cryptographic objects and related data used to support secure communications over the internet. Credentials may consist of public/private key pairs, symmetric keys, X.509 public key certificates, attribute certificates, and/or application data. Secured credentials are a set of one or more credentials that have been cryptographically secured, e.g. encrypted/MACed with a passkey. Credentials may be used in any end user device that can be connect to the Internet, such as:

- Desktop or laptop PC
- Mobile phone
- Personal digital assistant (PDA)

Since credentials usually contain secret information, they must never be sent in clear text and they should be stored securely. Using SACRED (Securely Available Credentials) protocols, users will be able to securely move their credentials between different locations, different Internet devices, and different storage media as needed. The IETF SACRED Working Group is working on the standardisation of a set of protocols for securely transferring credentials among devices.

Using security modules will solve many goals of the SACRED WG, but according to [13] hardware tokens are not appropriate for every environment. However, SACRED protocols can also complement hardware credential solutions by providing a standard mechanism for the update of credentials which are stored on the hardware device.

3.6.4 EToken

eToken is a portable USB device the size of an average house key, which offers a cost-effective method for authenticating users when accessing a network, and for securing electronic business applications. Just like in RSA SecureID authentication, an eToken is based on two-factor authentication. The user authenticates itself to the eToken by entering a PIN when the eToken is connected to a USB port of some device that contains a keyboard. Once the user has authenticated itself to the eToken, the eToken then authenticates itself to a network server on behalf of the user. Two different eTokens exist; eToken R2 and eToken PRO. eToken R2 contains secured and encrypted EEPROM memory chip and eToken PRO contains a smart card chip with ITSEC LE4 security. Other portable USB devices similar to eToken also exist, for example Rosetta USB [14].

USB tokens have the advantage that no card reader is required because modern information and communication equipment (like PCs, Laptops or future mobile phones) have a fast and easy to use USB connector with drivers and other supporting software. There is an ongoing IST project IST-1999-20323 SMART_USB [9], that is working on this task.

3.7 References

[9] http://www.smart-usb.org/index.htm

[10] Ross J. Anderson, Markus G. Kuhn: *Tamper Resistance - a Cautionary Note*, The Second USENIX Workshop on Electronic Commerce Proceedings, Oakland, California, November 18-21, 1996, pp 1-11, ISBN 1-880446-83-9

[11] RSA security inc., http://www.rsa.com

[12] iButton Homepage, http://www.iButton.com

[13] "Securely Available Credentials – Requirements", RFC 3157

[14] http://www.spyrus.com

4 Requirements on security modules

The different SHAMAN workpackages identified several high-level requirements on security modules in the course of their work, which are listed in the following. The requirements originate from heterogeneous access, distributed terminals and PKI as well as from requirements directly generated from smart card technology itself.

4.1 Requirements from workpackages WP1, WP2, WP3

From other work packages basic requirements on security modules have been collected. In summary, the detailed analysis produced the following list of requirements and most desirable features of security modules:

- Support of advanced security mechanisms, thus offering high security levels for authentication and session key agreement.
- Provision of a single SM implementing all the security mechanisms that are necessary to authenticate and agree session keys with multiple access networks.
- Providing assistance to the terminal in verifying the integrity of downloaded software (e.g. by MAC functions, integrity keys) which may be used to verify signatures on downloaded software.
- Verifying signatures on software that is downloaded to the SM.
- Protecting sensitive information such as secret keys against unauthorised access via physical or electrical interfaces.
- In the case where the user of a terminal abuses the services or the access that was granted from the network operator there should be an unique identification of the user for accountability purposes.
- Secure storage of a unique identifier for users which cannot be modified.
- Secure protection against disclosure or modification of network information and credentials on the SM by unauthorised network operators and other parties.
- Allowing the authorised users to access and modify stored personal data.
- Allowing the LEA to have access of the user’s personal stored data upon request based on a lawful authorisation.
- Mechanisms to switch on and off the PAN Manager functionality, whenever the owner of the PAN component wants it.
- The SM should be able to authenticate the PAN component user.
- Store authorisation information for different components.
- Provide mechanisms for security of PAN internal communications: integrity, confidentiality, authentication.

The requirements from WP3 (PKI environments) are more specific to the special requirements of PKI systems, since concrete functionality has to be provided for PKI. The basic requirements are:

- Safely generate, store and manage private keys.
- Manage the corresponding public keys.
- Handle certificates, policies and interoperability.
- Perform basic crypto functions based on stored keys.

Later work on SM requirements will rank these requirements and decide which individual functionality will be included in a reference architecture.

4.2 General requirements

Besides the requirements on SM technology from the heterogeneous access, distributed terminal and PKI work, further requirements are identified directly on the security modules (for example by local law) or generated from the SM technology itself.

- Comply with laws and regulations within the appropriate jurisdiction.
- Interoperation with mobile equipment as well as with other SMs.
- Exchangeable and replaceable SM implementation as an important feature of SM.
- Fixed device implementation for special usage of SMs (especially for low cost devices).

4.3 Requirements directly generated by the user

Direct requirements from the end user on the security module have not been analysed up to now. Such requirements might include a direct interface for monitoring the security module activity, they might include dual use of security modules for security functionality other than for telecommunications, or even for functionality other than security, for example notebook functionality. This investigation is for further study and requires a clearer understanding about how security modules are to be handled in the future.

5 Functionality of security modules required by the SHAMAN workpackages

In this section, the functionality required for a SM is deduced from the scenarios developed in SHAMAN work packages 1-3. For a more detailed description, see [15].

5.1 For WP1: Network access security

In WP1/D02 [21] requirements for securing the access link between a mobile node and a heterogeneous network were presented. The mechanisms required to be supported by the mobile node (= terminal equipment + security module) can be split into two:

- authentication and session key agreement;
- protection of the access link using the agreed session key.

In D05[1] the requirements on network access security that relate to the SM were discussed. In this document we take this further and consider how the data and functionality required to support these mechanisms should be split between the terminal equipment and the security module (SM). Regarding authentication and session key agreement, mechanisms based on secret key and public key techniques are investigated separately.

5.1.1 Authentication using secret key techniques

AAA for IPv6 is a candidate authentication and key agreement protocol based on secret key techniques which is investigated by WP1. The protocol is based on the authentication protocol from “draft-perkins-aaav6-05” [16] incorporating key distribution ideas from “draft-ietf-aaa-diameter-mobileip-09” [17]. For message flows and functionality splits, see Annex A of D08[2].

The choice of algorithms F1, F2 and F3 depends on the security association shared between the security module and the home network. The HMAC_MD5 algorithm using Ka as the key is suggested to be used for F1 and F2 in [16], whereas a simple hash using the MD5 algorithm applied to Ka plus the other algorithm inputs is suggested to be used for F3 in [17].

In [16] replay protection is provided using either random challenges or timestamps. In this document we only consider the use of random challenges.

Message flows for two versions of the protocol are specified:

- Basic protocol
- Basic protocol with an additional Home Network (HN) challenge

If the additional HN challenge is unavailable at the start of the protocol run then the terminal equipment (TE) must request it from the HN first. This requires an extra round trip between TE and HN.

Regarding the functionality split, it is suggested that it is feasible to perform all authentication and key agreement functions at the mobile node side on a smart card based security module. This requires two commands to the card. There are two variants of these two commands depending on whether the extra HN challenge is used - the optional N_4 and N_4’ parameters being included in square brackets.

- User authentication command:
    - input: N_1;
    - output: N_2, [N_4’,] ID, AUTN1
- Network authentication and session key agreement command:
    - input: N_3, [N_4,] AUTN2;

  - output: Ks

It should be noted that both versions of the protocol are stateful as far as the security module is concerned. In particular, in the basic and enhanced versions the security module is required to store the N_1 and N_2 parameters from the first command since these are used to verify AUTN2 send by the network when processing the second command. Furthermore, in the enhanced version of the protocol, the N_4' value must be stored from the previous run of the protocol.

5.1.2 Authentication using public key techniques

Several protocols based on public key techniques have been investigated as candidate authentication and key agreement protocols for network access.

The IKE protocol (Internet Key Exchange) is specified in Internet RFC 2409 [18]. It is a method to set-up Security Associations (SA). IKE is typically used to set-up SAs for IPsec but it may be used to set-up SAs for other security services.

In WP4, the IKE protocol has been investigated in detail with respect to security critical data and the functionality split between security module and terminal. This is described in D08[2] (chapter 3 and the annex).

The IKE protocol is today regarded as heavyweight and difficult to implement. Therefore, different candidate protocols are being investigated as possible successors to IKE. One such candidate is the JFK ("Just Fast Keying") protocol. Section 7.1 contains a description of the functionality split between security module and terminal for the JFKi protocol. Some functionality split options will be implemented in the demonstrator of SHAMAN WP5.

5.2 For WP2: Distributed terminals

In WP2/D03 [3] security requirements for distributed terminals have been presented. Some of these requirements are SM related and those are listed in document D05. SM related requirements in D05 [1] are divided into two classes depending on whether they are relevant for PAN internal or PAN external communication (or both). In this section we are focusing on PAN internal communications and secure download and our goal is to show how SMs can be implemented that fulfil these requirements.

Secure PAN internal communication means that communication between a trusted PAN component is encrypted, authenticated and integrity protected. Every pair of trusted components must have the ability to do mutual authentication. Two types of trusted components can exist in a PAN depending on whether the component is a first party component or a second party component. A first party component is a component which belongs to the owner of the PAN, whereas a second party component is a component which is including within the PAN communications but which does not belong to the PAN owner. Furthermore it was required that every first party component must have the ability to distinguish a first party component from a second party component. Two first party components **must** have the ability to make secure and authenticated key exchange without any user interaction.

Both symmetric and asymmetric cryptographic techniques use secret/private keys for authentication. If secret keys are stored in a PAN component, then they should be stored on tamper-resistant memory, which can be considered to be part of the SM. Note that PAN components do not necessarily need to store secret data, for example, components can base their authentication procedure on passkeys that can be human memorable.

Our goal in this section is to present the needed resources in SHAMAN compliant SMs for protecting PAN internal communication. SM resources depend on key management techniques, PAN scenarios from WP2 and the used cryptographic algorithms.

5.2.1 SM requirements on component initialisation

The wireless link between PAN components is inherently insecure, therefore first party components must have a link that has been secured either physically or cryptographically. Physical security is possible for example by using fixed connection, human involvement, low power channel. Physical techniques are not very flexible and often requires active user involvement. A better solution is to put cryptographic information in components to personalise them and make authentication possible - this procedure is called imprinting.

WP2 provides two imprinting mechanisms; one uses a manual authentication protocol (MANA) and it is used for imprinting a symmetric system the other one uses an imprinting mechanism for an asymmetric (public key) system.

In the first imprinting mechanism parties exchange Diffie-Hellman values g^x and g^y, where g is a generator of the group that is used in the Diffie-Hellman key-exchange and x and y are randomly chosen integers. The device then executes the MANA protocol for data $D=(g^x, g^y, \text{text})$ where text is any additional data (e.g., each other's identities) that the components may want to agree upon. MANA will provide that the devices are mutually authenticated and that they have the shared key $K=g^{xy}$. The required resources for the SM are:

- Pseudorandom generator (recommended).
- Computation of Diffie-Hellman exponentiation (recommended).
- Derivation of initial secret from shared Diffie-Hellman secret (recommended).
- Computation of one-way function (recommended)
- Storage of key (mandatory)

It is recommended that no sensitive information about the key be leaked out of the SM at any time. Therefore it is recommended that SMs inside the components choose integer x, y and compute Diffie-Hellman values g^x and g^y. A one-way function is needed for deriving further key material.

In the second imprinting mechanism the PAN component will get a private key and a public key that has been certified. Two approaches exist for key generation; the private key is either generated in the component or it is generated externally. For the SM there is a possibility that the private key is generated in the component, but not in the SM. From the SM’s point of view that case is just like any other case where the private key is generated externally.

Needed resources for SM are:

- Private key generation (if private key is generated in the SM) (mandatory)
- Pseudorandom generator (mandatory)
- Public operations (optional)
- Private key operations (mandatory)

A pseudorandom generator is needed for creating a private key.

5.2.2 SM requirements on subsequent authentication

As a result of initial authentication, the respective parties will have strong authentication keys. According to proposals in WP2 these keys will also be used in subsequent authentication unless the keys have expired or are not usable for other reasons. Therefore the authentication algorithm must reside on the SM.

5.2.3 SM requirements on integrity protection

According to the document D03[3] in PAN both data and signalling channels should be integrity protected. As currently proposed in WP2 integrity protection should rely on the usage of MACs instead of only encryption. From an SM point of view there are now two possibilities:

- The integrity key will be derived in SM (mandatory)

In this case the SM should have the ability to compute an integrity key as an output of a one-way function from the secret that has been installed on the SM. After derivation the SM can provide the integrity key to the PAN component, that can then compute the MACs.

- MACs will be computed in SM (mandatory)

The SM should have the capability to compute MACs. This is essentially the same requirement that the SM should be capable to compute values of a one-way function. However this method is inefficient because it will require that huge amounts of data are sent to and from the SM.

5.2.4 SM requirements on confidentiality protection

According to deliverable D03[3] both PAN data and signalling channels should be confidentially protected. As in integrity protection two possibilities exist:

- The confidentiality key will be derived in SM (mandatory)

In this case the SM should have the ability to compute a confidentially key as an output of a one-way function from the secret that has been installed on the SM. After derivation the SM can provide the confidentially key to the PAN component, which can then encrypt/decrypt communication data.

- Encryption/decryption is done in SM (mandatory)

This approach will require great computation power and fast I/O capabilities for the SM. This kind of solution may not be possible if the SM is implemented according to smart card specifications like ISO-7810 [19]. However, smart card technology can provide faster I/O capabilities for example, using the communication protocol T=1 (asynchronous half-duplex block transmission). Instead of serial interface also PCMCIA and USB interfaces can provide desired I/O capabilities.

5.2.5 SM requirements on identity and location privacy

When a connection between trusted PAN components is established then these components will mutually authenticate each other. This authentication procedure should not reveal a PAN component's identity to outsiders.

In WP2 a challenge/ response based authentication protocol is being investigated. Firstly, the service provider component SP sends a random challenge RAND to the service requestor component, which then replies with the expected response XRES and randomised hash of ID to the SP. This XRES is calculated from the RAND by using a one-way function and the shared-secret, thatsecret that has come from the imprinting process or is further derived from it. SP will search ID's in its local database and computes randomised hash values until it match to string that was given. After ID is found, SP use corresponding key to compute value RES and compares it to XRES.

The SM can derive the RES from the shared secret and RAND, or the XRES can be derived by an entity outside the PAN if the key that has been derived from initial-secret from imprinting is available to that entity. In both cases a one-way function in SM is needed.

5.2.6 Secure download

Components in a PAN may have different resources and therefore it is desired that the components can share these resources. Assume a case where none of the PAN components have a required functionality. In this situation one component can download an executable, which provides the functionality and then further distributes it to other components. Resource sharing and downloading executables from outside the PAN requires policies, which indicate the resources that a particular component can have access to.

Policies and access control mechanisms do not place any additional requirements on the SM, because if policies require that communication should be secured, the methods in sections 5.1 and 5.2 can be used. An additional requirement on the SM comes from the fact that origin and integrity of the message must be verified. SMs may contain or get symmetric secret keys for origin and integrity verification and the SM should do these verifications itself or derive further key material using a one-way function which is passed to the component to allow it to do the verifications instead. If public key techniques are used for verification, the SM may contain the public key of the root CA, because it can then verify the downloaded certificate, which should be provided along with the downloaded content. By using this certificate the SM may verify the signed digest of the content.

The requirements on the SM are:

- Storage of root CA's public key (optional)
- Computing digest of the content (optional)
- Verifying signature of the digests of the content (optional)

The SM can verify the integrity of the downloaded content if the digest of it is computed in the SM. However, this consumes a lot of the SM's resources.

5.2.7 Conclusions

It is required that all first party components have the ability to authenticate each other and make a secure key exchange without any user action. This goal can be achieved by imprinting these components. The result of imprinting determines the shared initial secret that must be stored on the SM. It is possible that components can be monitored at the time when this initial secret between the two components is generated if carried out on a non-tamperproof device, therefore it is recommended that secrets be generated in the SM. The challenge-response algorithm should be protected against replay attacks. The same key must not be used every time, therefore the SM should provide a mechanism for deriving a new key for the challenge-response algorithm.

Once the components are imprinted, the shared initial secret is used to secure the communication between the components. It is required that the SM contains the authentication algorithm. The one-way function is needed for deriving the key material for the encryption and integrity algorithms and the challenge/response algorithm for identification and location privacy.

Secure download requires that the origin and the integrity of downloaded content must be verified. The SM may verify the certificate and the signature of the digest of this content. If the integrity of downloaded content is verified in the SM, then the digest of content must also be computed in the SM.

5.3 For WP3: Personal PKI

We consider the security module requirements associated with the use of a 'Personal PKI'. We divide this discussion into three main parts:

- *Mobile device requirements*, i.e. the PKI requirements for a security module used with a 'client' mobile device (i.e. one which is a client of the Personal CA);
- *Personal CA requirement*s, i.e. the PKI requirements for the security module used by the personal CA device.
- *General requirements*, i.e. the PKI requirements applying to both Personal CA and client device.

5.3.1 Mobile device (client) requirements

The PKI-support requirements for a 'client' mobile device (i.e. one which is a client of the Personal CA) are as follows.

1. Key pair generation, i.e. the secure generation of one or more asymmetric key pairs for the mobile device;
2. Private key secure storage, i.e. the provision of facilities to store the mobile device private key(s) in a way which preserves its(their) confidentiality and integrity;

3. Root public key storage, i.e. the provision of facilities to store the public key(s) of the Personal CA(s) in a way which preserves their integrity;
4. Secure computation with device private key, i.e. the performance of cryptographic computations using the mobile device private key in a way that does not threaten its confidentiality or integrity;
5. Public key revocation handling (e.g. verifying CRLs or OCSP status messages supplied by the Personal CA).

It would certainly be highly desirable for requirements 1, 2 and 4 to be met by the security module for the mobile device. Meeting requirement 3 would also be desirable, although there exists the possibility that this and other information requiring integrity protection could be stored externally to the security module and integrity protected using a MAC computed using a key stored internally to the module. Requirement 5 is probably not something that needs to be handled within the SM.

5.3.2 Personal CA requirements

The PKI requirements for the Personal CA device are as follows.

1. Key pair generation, i.e. the secure generation of one or more asymmetric key pairs for the Personal CA;
2. Private key secure storage, i.e. the provision of facilities to store the Personal CA private key(s) in a way which preserves its(their) confidentiality and integrity;
3. Secure computation with Personal CA private key, i.e. the creation of public key certificates and, possibly, CRLs, using the Personal CA private key in a way that does not threaten its confidentiality or integrity.

The support for all three of these functions by a Personal CA security module would appear to be a high priority. Other possible functions for a Personal CA security module include the following:

1. Support for secure auditing, i.e. the support of integrity-protected records of all security-critical events (e.g. key generation, certificate creation, etc.);
2. Support for signed certificate status management messages, e.g. CRLs, or as required by OCSP.

5.3.3 General requirements

The exact set of cryptographic functions which need to be performed within the module as opposed to externally is negotiable. For example, in some circumstances it may be acceptable to perform part of the message processing for a signature computation externally to the secure device – how acceptable this is depends very much on the design of the cryptographic primitive. For example, the PSS-R based signature scheme in ISO/IEC 9796-2 [20] has been designed to permit some processing externally to a secure device without endangering the private key – this is not the case for all signature schemes.

All PKI-supporting security modules will need to be equipped with an authorisation function, to prevent unauthorised generation of new key pairs (and associated erasure of old key pairs).

5.4 Distribution of security

In D05 section 6.3, it was explained that SM might have limited resources. This resource problem can be solved in some cases by using distributed security. Our goal in this section is to present how SM or security services of SM can be distributed.

5.4.1 Distributed SM model

In D05 it was explained that it might be possible that a group of SMs could be connected to each other. Individual SMs can be considered to be security "atoms" and a group of SMs form a security "molecule". In the distributed SM model similar scenarios like in the case of a PAN arise, i.e., the centralised scenario where all SMs (atoms) are connected to a central SM or a fully-meshed scenario where all SMs are connected to each other. Different kinds of SMs with different resources/properties could be specified. However, for simplicity we only consider the case where all SMs have the same resources/properties.

In the distributed SM model task management is an important issue. The first possibility is that SMs gets their orders on what they should do from a PAN component and this PAN component must be aware of what all other SMs are doing. This first possibility will require the PAN component that is task manager to have some specific role in the PAN, for example it is the PAN manager component. The second possibility is that the task management is done by SMs only. Because all SMs have the same resources, they all must be capable to do task management. A different question is whether all SMs should participate in task management.

Security of the distributed SM model is analogous with security of PAN internal communications. Both data and signalling (related to task management) between SMs should be authenticated, confidential, integrity protected and perhaps anti-replay protected.

The distributed SM model can provide scenarios where some SMs can generate keys to other SMs and then distribute these keys in a secure way to these other SMs.

5.4.2 Distributed RSA

A public key operation requires a lot of resources; therefore in some situations it may be good that public key cryptographic primitives like RSA encrypt/sign operations are distributed. Many of the big CAs such as Verisign, Baltimore and Entrust provide certificates for RSA public keys and actually RSA is the only public key cryptographic system that is supported by these CAs. So RSA can be considered as the standard in public key cryptography.

In WP4, a method for distributing RSA operations between a tamper resistant device (TRD) and an insecure device (non-tamper resistant device) was worked out. A very rough estimate is that this protocol takes about 7% time compared to the usual method. It is described in detail in D08[2], chapter 3.

5.4.3 Conclusions

The SM has limited resources; therefore it may have to distribute the workload. Two approaches have been presented, the first one is to distribute the SM and second one is to distribute security functionality between the SM and terminal.

The distributed SM model has following advantages

- Parallel computing between SMs is possible
- Different SMs may share secret data, like secret keys.

However the distributed SM has following disadvantages

- High I/O capabilities are needed
- Task management between SMs is problematic
- Connections between SMs must be secured, and this consumes SMs resources.

It has been shown how to distribute securely the IKE protocol between a SM and a terminal. It is identified that a single private key operation can be a complex task for a SM. A mechanism for distributing security for RSA private key operations has been presented. In case of 1024-bit RSA keys, the workload of computation for a SM is only 7% compared to standard RSA. However, using the distributed RSA method requires four messages to be transferred between the SM and terminal.

The distribution of RSA does not prohibit plastic roaming, although some pre-computed operations have to be carried out in this case.

5.5 References

[15] ‘Intermediate Report: Results of Review, Requirements and Reference Architecture‘,Deliverable D05, IST-2000-25350-SHAMAN

[16] P. Flykt, C. E. Perkins, T. Eklund, AAA for IPv6 Network Access, Internet Draft, draft-perkins-aaav6-05, March 2002.

[17] P. R. Calhoun, T. Johansson, C. E. Perkins, Diameter Mobile IPv4 Application, Internet Draft, draft-ietf-aaa-diameter-mobileip-09, March 2002.

[18] D. Harkins, D. Carrel, The Internet Key Exchange (IKE), Internet RFC 2409.

[19] ISO/IEC-7810 , Identification cards, physical characteristics

[20] ISO 9796-2, Digital signatures giving message recovery, Part 2: Mechanisms using a hash function

[21] ‘Intermediate Report: Results of Review, Requirements and Reference Architecture’, Deliverable D02, IST-2000-25350-SHAMAN

6 A SHAMAN security module reference model

This section serves as a summary and reference and can be used as a basis for an implementation of a SHAMAN-compliant security module.

Several possible versions of the SM are introduced, qualified by their “security level”, providing different basic functions and protocol support; these notions are defined in the following section.

In section 6.2 the required functions for each of the security levels are presented.

6.1 Notions used in the reference model

6.1.1 Security levels

The importance of the SM features which have been worked out in section 5 differ from each other and depend on the desired level of security.

We define 3 levels here:

1. Intermediate
2. High
3. Personal-CA-enabled

The basic functionality which should be implemented on them is given in section 6.2.

“Intermediate” is the lowest security level (the term “low security” is not used here!). It supports only the functionality which is required as minimum. Whether this level is sufficient or not is always debatable.

“High” refers to the highest level of security. It fully includes the intermediate level functions, and provides some additional functions for higher security, but also convenience.

“Personal-CA-enabled” denotes a SM which includes all the functionality that is already included in the “High”-level and – additionally – enables the SM to be used in a personal PKI (for a client of a personal CA and for the CA device itself). The reason that this level was introduced is that some of the features for personal CAs require the SM to have considerable performance and storage capacity which may not be available with today’s devices.

6.1.2 Protocol support

A SHAMAN-compliant SM must support symmetric key and/or asymmetric key protocols for authentication and key agreement. These protocols may be used for secure network access (WP1) or secure PAN communications (WP2). Examples of symmetric key protocols for secure network access include the AAA for IPv6 protocol [16], [17], while examples of asymmetric key protocols include IKE or son-of-IKE protocols [18].

An SM like a smart card usually provides a defined command set. Each command reads a number of input parameters and returns a value which corresponds to a basic function like “hash”, “digital signature” etc. In order to implement the above-mentioned protocols, this will be, in general, not be enough: Some state information may have to be maintained or other (eventually more complex) commands must be added to the SM’s command set.

6.2 Functionality for the different levels

6.2.1 Intermediate level

The following basic functions are required:

- Pseudo-random number generator

- Storage of long-lived keys
- Private key operations (signature, decrypting)
- One-way function (for derivation of integrity or confidentiality keys, usage in the AAA protocol etc.)

Concerning the SHAMAN protocols: the intermediate-level SM is only expected to provide basic support for authentication and key agreement protocols. For example, an intermediate-level SM might be expected to support the following :

- AAA protocol for IPv6, basic protocol (see [16])
- Distributed IKE, authentication with pre-shared key, simpler scenario (see [18])

Study of the functionality split between the SM and the terminal for other protocols is included in section 7.

6.2.2 High level

In addition to the intermediate level, the following functions shall be provided:

- Storage of short-term keys (session keys, e.g.)
- Computation of Diffie-Hellman secret / derivation of initial secret from DH secret
- On-card generation of a key pair
- Storage of security context data (for plastic roaming)

Concerning the SHAMAN protocols: the high-level SM is expected to provide enhanced support for authentication and key agreement protocols. For example, a high-level SM might be expected to support the following :

- AAA protocol with additional HN nonce
- Distributed IKE, authentication with pre-shared key, complicated scenario
- Distributed IKE, authentication with signatures (either scenario)

Study of the functionality split between the SM and the terminal for other protocols is included in section 7.

6.2.3 Personal-CA-level

The following additional functions are needed:

- Storage of certificates
- Creation of certificates
- Secure auditing

Creation of certificates, although not required from the WP3 viewpoint, is a natural task for CAs. Current implementations (like the WIM [4], e.g.) already offer this possibility.

Secure auditing, i.e. the support of integrity-protected records of all security-critical events (like key generation, certificate creation, etc.) is not a high-priority WP3 requirement; however, it is desirable for any CA to have an overview about the certificates it has created in the past.

6.3 Hardware

In the following, hardware specifications for the realisation of a SHAMAN security module are given. This is one possible way to realise a SHAMAN security module, although it should be noted that there may be other possibilities.

The electronic signals and transmission protocols of the SHAMAN smart card shall be in accordance with ISO/IEC 7816-3 [22]. The implementation of the T=0 protocol is mandatory, T=1 is optional.

Given the requirements for the different security levels, conclusions for the necessary storage capacity and processor properties can be drawn. Long-term keys, e.g., usually have a size of 1024 bit, certificates typically a few kB. This data must be stored in the EEPROM of the card. For the intermediate level an EEPROM size of 16 kB will therefore probably be sufficient, but for high level at least 32 kB is recommended.

The SHAMAN smart card will also require one or more coprocessing units. First, there will be a numerical processor which only needs to provide a few basic calculations, like the exponentiation and modulo division of big numbers, as required for the Diffie-Hellman protocol, for example. Second, one or more specialised units are necessary in order to reduce processing time, for example hard-coded random number generators, hash calculators or RSA units, as required for the intermediate security level. A dedicated DES unit is not required because the symmetric algorithms using short-term keys will be performed outside the security module.

6.4 Software

The SHAMAN reference model does not mandate the use of a certain operating or file system on the card. For interoperability, only the APDU format would have to be fixed. This is a possible task for the future; the exact format is outside the scope of this document.

In general, the card should support the common commands according to the ISO 7816-4 standard. As for other cards in the telecommunication business (like SIM cards), the class byte for all APDUs should be ‘A0’. For the SHAMAN-specific protocol support, some additional APDUs will have to be introduced. The exact number and format of those APDUs depends on the details of the functionality split the SHAMAN security module is supposed to implement. As an example, 2 APDUs must be specified for the JFK protocol (see 7.1), one for triggering the smart card to send a nonce and a Diffie-Hellman value, and another one which contains data for the smart card to sign and return and which tells the smart card to create a number of short-term secrets.

The security module must provide the cryptographic algorithms which are mandated by the protocols it supports. Those are given in the following list:

1. One-way function: SHA-1
2. MAC: HMAC [23]
3. Signature: RSA (key length of 1024 bit)
4. Diffie-Hellman: as specified in JFK, support at least one of the groups MODP1, MODP2, EC2N3 or EC2N4; MODP1 is mandatory [18]

6.5 References

[22] ISO 7816-3, Integrated circuit cards with contact, Electronic signals and transmission protocols

[23] Krawczyk et al.: HMAC, Keyed Hashing for Message Authentication, Internet RFC 2104.

7 Functionality split of selected protocols

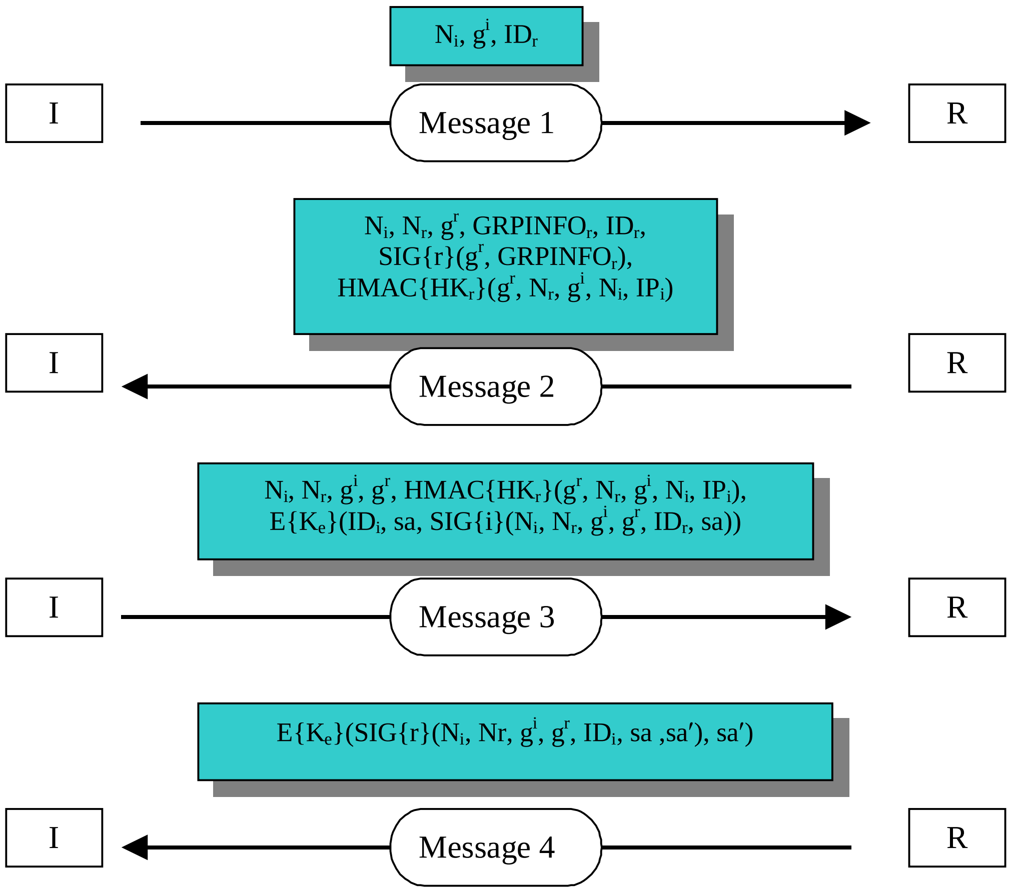
7.1 Functionality split of JFKi initiator functions between SM and ME

7.1.1 JFK overview

Just Fast Keying (JFK) [24] is an alternative to the Internet Key Exchange (IKE) protocol [18]. JFK is less complex than IKE, has fewer roundtrips and is designed to be more robust against DoS attacks. In this document we consider the use of the JFK protocol between a mobile node and a network.

There are two entities involved in the JFK protocol: the initiator and the responder. Furthermore there are two variants of the JFK protocol denoted JFKi and JFKr. The variants are very similar in many respects, but there are some differences: JFKi provides active identity protection for the initiator and no identity protection for the responder, whereas JFKr provides active identity protection for the responder and passive identity protection for the initiator. Both variants provide resistance to memory and CPU exhaustion DoS attacks against the responder. In our scenario we assume that the network acts as the JFK responder because it is considered more important to provide denial of service protection to the network side. Furthermore, as identity confidentiality for the mobile node is more important than identity confidentiality for the network, the JFKi variant is selected.

Figure 1 illustrates the messages exchanged in the JFKi protocol. The mobile acts as the initiator (I) and the network as the responder (R).


N_i, g^i, ID_r
I
Message 1
R
N_i, N_r, g^r, GRPINFO$_r$, ID_r, SIG{r}(g^r, GRPINFO$_r$), HMAC{HK_r}(g^r, N_r, g^i, N_i, IP_i)
I
Message 2
R
N_i, N_r, g^i, g^r, HMAC{HK_r}(g^r, N_r, g^i, N_i, IP_i), E{K_e}(ID_i, sa, SIG{i}(N_i, N_r, g^i, g^r, ID_r, sa))
I
Message 3
R
E{K_e}(SIG{r}(N_i, Nr, g^i, g^r, ID_i, sa ,sa′), sa′)
I
Message 4
R

The following notation is used:

- $E\{K\}(M)$: Encryption of M with secret key K.$HMAC\{K\}(M)$: Keyed hash of M using key K in an HMAC scheme.
- $SIG\{x\}(M)$: Digital signature of M using the private key belonging to x.
- g^x: Diffie-Hellman exponentials, also identifying the Diffie-Hellman group.
- g^i, g^r: Initiator and responder exponentials.
- N_i N_r: Initiator and responder nonces. Fresh for each protocol invocation.
- IP_i: The initiator's network identifier (IPv4 or IPv6 address).
- *sa*: Defines the cryptographic and other properties of the Security Association (SA) the initiator wants to establish. It contains a Domain-of-Interpretation, which JFK understands, and an application-specific bit-string.
- *sa*': Any information that the responder needs to provide to the Initiator with respect to the application SA (e.g., the responder's SPI, in IPsec).
- HK_r: A transient hash key private to the responder, which changes periodically: the responder must pick a new g^r every time HK_r changes.
- K_{ir}: A shared key derived from g^{ir}, N_i, and N_r, used as part of the application SA (e.g., IPsec SA).
- K_e: A shared key derived from g^{ir}, N_i, and N_r, used to protect messages (3) and (4) of the protocol.
- ID_i, ID_r: Initiator and responder certificates or public-key identifying information.
- $ID_{r'}$: An indication by the initiator to the responder as to what identity (and corresponding key material) the latter should use to authenticate to the former.
- $GRPINFO_r$: A list of all Diffie-Hellman groups supported by the responder, the symmetric algorithm used to protect messages (3) and (4), the hash function used for key generation and the signature scheme.

The key used to protect messages (3) and (4), K_e, is computed as $HMAC\{g^{ir}\}(N_i, N_r, 1)$. The session key used by IPsec (or any other application), K_{ir}, is computed as $HMAC\{g^{ir}\}(N_i, N_r, 0)$. Key K_s, used for lightweight mode (see below), is computed as $HMAC\{g^{ir}\}(N_i, N_r, 3)$. A full description of the protocol and the notation used is contained in section 2.5.5.1.1 within D09.

For rekeying there are two options: the full JFK protocol can be run again or a lightweight mode can be run where the expensive public key signatures in messages (3) and (4) are replaced with HMAC operations. The lightweight mode of the protocol is identical to the full protocol except that the signature in message (3) is replaced with $HMAC\{K_s'\}(N_i, N_r, g^i, g^r, ID_r, sa)$ and the signature in message (4) is replaced with $HMAC\{K_s'\}(N_i, Nr, g^i, g^r, ID_i, sa ,sa')$. The key K_s' is the K_s computed during the previous run of the protocol.

To prevent linking of protocol exchanges involving the same mobile node which would compromise user identity confidentiality, we assume that the MN changes g^i for each protocol run.

7.1.2 Assessment of security functions at the initiator

In this document we assume that certain critical security functions at the mobile node can be performed on a tamper resistant device (TRD) such as a smart card. Therefore we consider the mobile node to be split into the two components: a mobile equipment (ME) and a tamper resistant device (TRD). To help identify which functions should be implemented in the TRD, we consider the security requirements on the following data which is used by the initiator:

- The Diffie-Hellman private exponent i.
- The Diffie-Hellman secret g^{ir}
- The initiator nonce N_i
- Private signature generation key – This is the private key used to generate *Sig{i}(…)*
- Public signature verification key – This is the public key used to verify *Sig{i}(…)*. In general the initiator may obtain the signature verification key by verifying a chain of public key certificates leading to a 'root' public key which the initiator trusts.
- K_e – This is an encryption key generated from the Diffie-Hellman secret. It is used to encrypt messages (3) and (4) in the protocol.
- K_{ir} – This is an encryption key generated from the Diffie Hellman secret. It is used as part of the application SA (e.g. IPsec SA).
- Security Association *sa* – Defines the cryptographic and other properties of the Security Association (SA) that the initiator wants to establish.
- Security Association *sa′* - Any information that the responder needs to provide to the initiator with respect to the application SA.

7.1.2.1 Diffie-Hellman private exponent i

The Diffie-Hellman private exponent i is used to generate the Diffie-Hellman secret g^{ir} which in turn is used to generate the keys K_e, K_{ir} and K_s. If the private exponent is generated and used within the TRD then its value need never be revealed outside the TRD. Never revealing the private exponent outside the TRD is only an advantage if there is a requirement for one or more of the keys K_e, K_{ir} and K_s never to be revealed outside the TRD. K_{ir} always has to be passed to the ME because the IPsec processing cannot be done on the TRD for performance reasons. However, there are situations where it might be advantageous to never reveal K_e or K_{ir} outside the TRD.

7.1.2.2 Diffie-Hellman secret g^{ir}

The Diffie-Hellman secret is used to generate the keys K_e, K_{ir} and K_s. If the generation of these keys is done within the TRD, then the Diffie-Hellman secret need never be revealed outside the TRD. Never revealing the Diffie-Hellman secret outside the TRD is only an advantage if there is a requirement for one or more of the keys K_e, K_{ir} and K_s never to be revealed outside the TRD. K_{ir} always has to be passed to the ME because the IPsec processing cannot be done on the TRD for performance reasons. However, there are situations where it might be advantageous to never reveal K_{ir} or K_e outside the TRD.

7.1.2.3 Initiator nonce N_i

The initiator nonce is used (among other things) to generate the keys K_e, K_{ir} and K_s. In this setting it is used to provide key independence when one or both parties reuse the same Diffie-Hellman exponential in successive protocol runs. To satisfy this requirement the nonce must be fresh for each protocol run, but it need not be random.

The initiator nonce also provides protection against pre-play attacks. In order to do this it must be a 'good' random number. If the nonce were generated by the ME and predictable then a pre-play attack would be possible if the initiator reused the same Diffie-Hellman exponent in successive protocol runs. The attack would require the attacker to have physical access to the target's TRD and find the last nonce generated by the ME. The attacker would then use the TRD to run the JFK protocol x times with the legitimate network using the next x initiator nonces. For each run of the protocol the attacker would store the messages (2) and (4) generated by the legitimate network. The attacker can now return the TRD to the target and masquerade as the legitimate network towards the target MN because he knows the correct message (2) and message (4) for the next x runs of the protocol.

To avoid such pre-play attacks it is necessary for either the initiator's Diffie-Hellman exponent or the initiator's nonce to be a 'good' random for each run of the protocol. If both these values are generated on the ME then there is a risk of the ME being tampered with or has a poor JFK implementation which may make it vulnerable to a pre-play attack. To avoid such attacks the Diffie-Hellman exponent and the initiator's nonce could be generated and used in the TRD. However, this does not add much security because a malicious ME implementation could simply not perform authentication of the (false) network, but incorrectly inform the user that the JFK protocol has been successful and that a secure IPsec connection has been established

In summary there seems little benefit in generating the nonce on the TRD. However, if the Diffie-Hellman private exponent has to be generated and used in the TRD for security reasons then it might be worthwhile to also do the nonce generation in the TRD to avoid duplication of functionality between the TRD and the ME.

7.1.2.4 Private signature generation key

The private signature key is a long term key used by the mobile node to sign message (3). If the private key is revealed to an attacker then he will be able to clone the mobile node. It is therefore highly advantageous for the private key to be securely stored and used within the TRD such that its value is never revealed outside the TRD.

To protect the private key the data to be signed could be passed to the TRD for signing. Alternatively the data to be signed could be hashed on the ME and the hash could then be signed by the TRD. This would save on TRD interface bandwidth and internal storage costs, but there are security concerns because it could allow an attacker to perform a chosen plaintext attack to help him discover the private signing key. Using this method the attacker would be able to substitute the legitimate hashes for hashes that he has chosen himself. To prevent this attack it is recommended that double hashing is performed where the ME will hash the message to be signed and pass the result to the TRD which then hashes the hash and signs the result. The re-hashing process on the TRD takes away the vulnerability of the chosen plaintext attack.

7.1.2.5 Public signature verification key

If an attacker could modify or replace the initiator's trusted root key then he could fool the initiator into accepting a signature from a false network. To mitigate against this attack all public keys could be stored in the TRD to prevent unauthorised modification and all certificate and signature verification could be done within the TRD to avoid having to reveal the pulbic keys to the ME where they might be modified. However, this would still not prevent an attacker modifying an ME such that it always accepts signatures from false networks without properly verifying the signature on the TRD. Therefore, there seems little benefit from a security perspective in storing and using the public verification keys within the TRD.

There might be other non security reasons for performing signature verification on the TRD. For instance, if the root public keys/certificates are provisioned on the TRD then it might be more efficient to carry out signature verification on the TRD rather than have to pass the keys / certificates to the ME for processing. However, if the keys/certificates are kept on the ME, then it would be more efficient to perform the signature verification on the ME.

7.1.2.6 The encryption K_e

It is only necessary to keep K_e secret to the TRD if the encryption and decryption of messages (3) and (4) is also done on the TRD. If messages (3) and (4) are encrypted and decrypted on the TRD then the identity of the initiator and details of the security association (but not the encryption key K_{ir}) are not revealed to the ME. However, the security association must be revealed to the ME anyway because the IPsec processing must be done on the ME for performance reasons. Furthermore, it is likely that the user identity needs to be known by the ME anyway. Therefore, there seems little advantage in keeping K_e secret to the TRD.

7.1.2.7 The negotiated secret key K_{ir}

K_{ir} must be revealed to the ME because the IPsec processing must be done on the ME for performance reasons. If this key is compromised, then it will allow an attacker to decrypt and hijack sessions. This risk can be reduced to an acceptable level if K_{ir} is changed frequently.

7.1.2.8 *Security Association* sa *and* sa′

The security association must be revealed to the ME because the IPsec processing must be done on the ME for performance reasons.

7.1.3 Functionality split options

In this section we define five options for splitting JFK initiator functionality between the ME and the TRD. These options are based on the results of the analysis performed in section 6.2. The options are listed in order of increasing complexity with respect to the TRD.

Option 1 – Signature generation on the TRD

Option 2 – Option 1 plus random number generation and exponentiation on the TRD

Option 3 – Option 2 plus HMAC operations on the TRD

Option 4 – Option 3 plus encryption/decryption on the TRD

Option 5 – Option 4 plus signature/certificate verification on the TRD[1]

The options describe the functionality split for the full JFK protocol. The functionality split for the lightweight mode is assumed to the the same except that the initiator signature is replaced with $HMAC\{K_s'\}(N_i, N_r, g^i, g^r, ID_r, sa)$ and the responder signature is replaced with $HMAC\{K_s'\}(N_i, Nr, g^i, g^r, ID_i, sa ,sa')$. The location of the HMAC operation is assumed to be the same as the location of the signature option for each of the options listed below.

7.1.3.1 Option 1 - Signature generation on the TRD

7.1.4 Message flow

ME→N:	N_i, g^i, ID_r
N→ME:	$N_i, N_r, g^r, GRPINFO_r, ID_r, SIG\{r\}(g^r, GRPINFO_r), HMAC\{HK_r\}(g^r, N_r, g^i, N_i, IP_i)$
ME→TRD:	$\boldsymbol{N_i, N_r, g^r, GRPINFO_r, ID_r, sa}$
TRD→ME:	$\boldsymbol{SIG\{i\}(N_i, N_r, g^i, g^r, ID_r, sa)}$
ME→N:	$N_i, N_r, g^i, g^r, HMAC\{HKr\}(g^r, N_r, g^i, N_i, IP_i), E\{K_e\}(ID_i, sa, SIG\{i\}(N_i, N_r, g^i, g^r, ID_r, sa))$
N→ME	$E\{K_e\}(SIG\{r\}(N_i, N_r, g^i, g^r, ID_i, sa, sa'), sa')$

7.1.5 Requirements on TRD

The TRD is involved in one step. It is required to perform one signature operation to generate $SIG\{i\}(N_i, N_r, g^i, g^r, ID_r, sa)$.

A variant of this option is that the data to be signed could be hashed on the ME and the hash could then be signed by the TRD. This would save on TRD interface bandwidth and internal storage costs, but there are security concerns with this approach (see section 6.2.4 above).

[1] Note that the signature/certificate verification could be done on the TRD independently of the other functions so in fact three other options exist where the signature/certificate verification on the TRD is added to options 1, 2 and 3, respectively.

7.1.5.1 Option 2 – Option 1 plus random number generation and exponentiation on the TRD

7.1.6 Message flow

TRD→ME:	**N_i, g^i, ID_r**
ME→N:	N_i, g^i, ID_r
N→ME:	N_i, N_r, g^r, $GRPINFO_r$, ID_r, $SIG\{r\}(g^r, GRPINFO_r)$, $HMAC\{HK_r\}(g^r, N_r, g^i, N_i, IP_i)$
ME→TRD:	**$[N_i]$ N_r, g^r, $GRPINFO_r$, ID_r, sa**
TRD→ME:	**g^{ir}, $SIG\{i\}(N_i, N_r, g^i, g^r, ID_r, sa)$**
ME→N:	N_i, N_r, g^i, g^r, $HMAC\{HKr\}(g^r, N_r, g^i, N_i, IP_i)$, $E\{K_e\}(ID_i, sa, SIG\{i\}(N_i, N_r, g^i, g^r, ID_r, sa))$
N→ME	$E\{K_e\}(SIG\{r\}(N_i, N_r, g^i, g^r, ID_i, sa, sa'), sa')$

7.1.7 Requirements on the TRD

The TRD is involved in two steps. In the first step it is required to generate random numbers i and N_i, and then perform one exponentiation to generate g^i. In the second step it is required to perform one exponentiation to generate g^{ir} and one signature operation to generate $SIG\{i\}(N_i, N_r, g^i, g^r, ID_r, sa)$.

In the second step, N_i could be omitted from the message from the ME to the TRD to save on TRD interface bandwidth costs if it was cached from the first step.

7.1.7.1 Option 3 - Option 2 plus HMAC operations on the TRD

7.1.8 Message flow

TRD→ME:	**N_i, g^i, ID_r**
ME→N:	N_i, g^i, ID_r
N→ME:	N_i, N_r, g^r, $GRPINFO_r$, ID_r, $SIG\{r\}(g^r, GRPINFO_r)$, $HMAC\{HK_r\}(g^r, N_r, g^i, N_i, IP_i)$
ME→TRD:	**$[N_i]$ N_r, g^r, $GRPINFO_r$, ID_r, sa**
TRD→ME:	**K_e, K_{ir}, $SIG\{i\}(Ni, Nr, gi, gr, ID_r, sa)$**
ME→N:	N_i, N_r, g^i, g^r, $HMAC\{HKr\}(g^r, N_r, g^i, N_i, IP_i)$, $E\{K_e\}(ID_i, sa, SIG\{i\}(N_i, N_r, g^i, g^r, ID_r, sa))$
N→ME	$E\{K_e\}(SIG\{r\}(N_i, N_r, g^i, g^r, ID_i, sa, sa'), sa')$

7.1.9 Requirements on the TRD

The TRD is involved in two steps. In the first step it is required to generate random numbers i and N_i, and then perform one exponentiation to generate g^i. In the second step it is required to perform one exponentiation to generate g^{ir}, perform three HMAC operations to generate K_e, K_{ir}, and K_s and one signature operation to generate $SIG\{i\}(N_i, N_r, g^i, g^r, ID_r, sa)$.

In the second step, N_i could be omitted from the message from the ME to the TRD to save on TRD interface bandwidth costs if it was cached from the first step.

7.1.9.1 Option 4 - Option 3 plus encryption/decryption on the TRD

7.1.10 Message flow

TRD→ME:	**N_i, g^i, ID_r**
ME→N:	N_i, g^i, ID_r

N$\rightarrow$ME:	$N_i, N_r, g^r, GRPINFO_r, ID_r, SIG\{r\}(g^r, GRPINFO_r), HMAC\{HK_r\}(g^r, N_r, g^i, N_i, IP_i)$
ME$\rightarrow$TRD:	**$[N_i] N_r, g^r, GRPINFO_r, ID_r, sa$**
TRD$\rightarrow$ME:	**$K_{ir}, E\{K_e\}(ID_i, sa, SIG\{i\}(N_i, N_r, g^i, g^r, ID_r, sa))$**
ME$\rightarrow$N:	$N_i, N_r, g^i, g^r, HMAC\{HKr\}(g^r, N_r, g^i, N_i, IP_i), E\{K_e\}(ID_i, sa, SIG\{i\}(N_i, N_r, g^i, g^r, ID_r, sa))$
N$\rightarrow$ME	$E\{K_e\}(SIG\{r\}(N_i, N_r, g^i, g^r, ID_i, sa, sa'), sa')$
ME$\rightarrow$TRD:	**$E\{K_e\}(SIG\{r\}(N_i, N_r, g^i, g^r, ID_i, sa, sa'), sa')$**
TRD$\rightarrow$ME:	**$SIG\{i\}(N_i, N_r, g^i, g^r, ID_r, sa)), sa'$**

7.1.11 Requirements on the TRD

The TRD is involved in three steps. In the first step it is required to generate random numbers i and N_i, and then perform one exponentiation to generate g^i. In the second step it is required to perform one exponentiation to generate g^{ir}, perform three HMAC operations to generate K_e, K_{ir}, and K_s, one signature operation to generate $SIG\{i\}(N_i, N_r, g^i, g^r, ID_r, sa)$ and an encryption operation to generate $E\{K_e\}(ID_i, sa, SIG\{i\}(N_i, N_r, g^i, g^r, ID_r, sa))$. In the third step it is required to perform a decryption operation to recover $SIG\{r\}(N_i, N_r, g^i, g^r, ID_i, sa, sa'), sa'$.

In the second step, N_i could be omitted from the message from the ME to the TRD to save on TRD interface bandwidth costs if it was cached from the first step.

7.1.11.1 Option 5 - Option 4 plus signature/certificate verification on the TRD

7.1.12 Message flow

TRD$\rightarrow$ME:	**N_i, g^i, ID_r**
ME$\rightarrow$N:	N_i, g^i, ID_r
N$\rightarrow$ME:	$N_i, N_r, g^r, GRPINFO_r, ID_r, SIG\{r\}(g^r, GRPINFO_r), HMAC\{HK_r\}(g^r, N_r, g^i, N_i, IP_i)$
ME$\rightarrow$TRD:	**$[N_i] N_r, g^r, GRPINFO_r, ID_r, sa$**
TRD$\rightarrow$ME:	**$K_{ir}, E\{K_e\}(ID_i, sa, SIG\{i\}(N_i, N_r, g^i, g^r, ID_r, sa))$**
ME$\rightarrow$N:	$N_i, N_r, g^i, g^r, HMAC\{HKr\}(g^r, N_r, g^i, N_i, IP_i), E\{K_e\}(ID_i, sa, SIG\{i\}(N_i, N_r, g^i, g^r, ID_r, sa))$
N$\rightarrow$ME	$E\{K_e\}(SIG\{r\}(N_i, N_r, g^i, g^r, ID_i, sa, sa'), sa')$
ME$\rightarrow$TRD:	**$E\{K_e\}(SIG\{r\}(N_i, N_r, g^i, g^r, ID_i, sa, sa'), sa')$**
TRD$\rightarrow$ME:	**sa'**

7.1.13 Requirements on the TRD

The TRD is involved in three steps. In the first step it is required to generate random numbers i and N_i, and then perform one exponentiation to generate g^i. In the second step it is required to perform one exponentiation to generate g^{ir}, perform three HMAC operations to generate K_e, K_{ir}, and K_s, one signature operation to generate $SIG\{i\}(N_i, N_r, g^i, g^r, ID_r, sa)$ and an encryption operation to generate $E\{K_e\}(ID_i, sa, SIG\{i\}(N_i, N_r, g^i, g^r, ID_r, sa))$. In the third step it is required to perform a decryption operation to recover $SIG\{r\}(N_i, N_r, g^i, g^r, ID_i, sa, sa'), sa'$, and a signature verification operation to verify $SIG\{r\}(N_i, N_r, g^i, g^r, ID_i, sa, sa')$. Note that in addition to the signature verification operation, the TRD may also be required to verify a chain of certificates to verify the authenticity of the signature verification key.

In the second step, N_i could be omitted from the message from the ME to the TRD to save on TRD interface bandwidth costs if it was cached from the first step.

7.1.14 Conclusion

Based on the assessment of security functions at the initiator and the description of the functionality split options, we now attempt to make some recommendations about which option or options should be selected in our particular setting for the JFKi protocol.

The safest approach would be to implement all security functions at the initiator side in the TRD. If this is feasible and cost effective, it is recommended to adopt this approach. However, for the purposes of this study we assume that it is significantly more expensive to perform security functions on the TRD compared to on the ME. Therefore we attempt to balance security against cost when deciding which security functions should be implemented on the TRD.

There seems little security benefit in doing expensive signature/certificate verification operations on the TRD compared to doing them on the ME, therefore it is not considered necessary from a security perspective to implement this functionality on the TRD[2]. Furthermore, there seems little benefit in doing encryption/decryption on the TRD because the initiator identity probably has to be revealed to the ME anyway, therefore this functionality can be implemented on the ME.

For the remaining functions, the only function which seems essential to implement on the TRD is signature generation. The other functions could be implemented in the ME but there are security advantages if they are implemented in the TRD. The selection of the most appropriate options to implement on the TRD depends largely on whether the lightweight mode is used for rekeying.

If the lightweight mode is used for rekeying then K_s must be kept confidential, otherwise it would be possible for an attacker to masquerade as the legitimate initiator during the rekeying protocol without having to obtain or clone the legitimate initiator’s TRD. If K_s, or parameters which may be used to derive it, are revealed to the ME, then it may be possible for an attacker to obtain K_s, which may be used subsequently to masquerade as a legitimate initiator. The attacker could obtain K_s by borrowing the ME or by planting a hostile application on the ME. It is therefore neccessary to protect the confidentiality of K_s by keeping it secret to the TRD. Only option 3 provides this protection.

If the full protocol is used for rekeying then there is no requirement to keep K_s confidential. However, it is still important to protect the confidentiality of the private signature key, otherwise it would be possible for an attacker to masquerade as the legitimate intiator. Option 1 provides protection for the private signature key. Option 2 seems to provide little benefit compared to option 1 in this scenario.

Clearly option 3 is more complex with respect to the algorithm support that have to be implemented on the TRD. On the other hand option 3 allows a more lightweight protocol to be used for rekeying. The final decision on whether to select option 1 or option 3 should be based on a trade-off between the complexity of implementation on the TRD and the desire for a lightweight mode for rekeying. This analysis cannot be done until further information on the performance of the protocol is obtained using TRDs based on real smart cards and a protocol excecution environment which is representative of a real mobile network. The results of the SHAMAN network access authentication demonstrator are therefore expected to be useful in helping to assess which functionality split option should be selected.

7.2 Authorisation protocols and SM

7.2.1 Introduction

WP2 contains a description of an authorisation protocol. In this protocol PAN components that have an RSA private key can delegate authorisation for other components to use this private key. Authorisation is done in such a way that the authorised party does not get any sensitive information about the private key, but it can still use this key.

Three roles for PAN components were identified. The master is the owner of the RSA private key, while the slave is the device that uses the master's private key and the server is the entity that helps the

[2] Note that there may be non-security reasons to do signature/certificate verification on the TRD as discussed in section 6.2.5.

slave use the master’s private key. Before the slave can use the master's private key an initialisation procedure must be executed. In the initialisation step the master splits its private key and gives one half-key to the slave and the other half-key to the server. After initialisation the slave sends messages that should be signed/decrypted to the server. The server applies its own half-key to messages from the slave and sends responses back to the slave. The slave will complete these messages by applying its own half-key to response messages.

In this chapter we describe the requirements for SM that comes from delegated authorisations. We classify requirements according to the desired security level. We use two security levels. These security levels are:

1 Intermediate

2 High

We explain in this chapter that a malicious slave is the most serious attacker for a server and a malicious server is most the serious attacker for a slave. Only SM with high level security protect against these threats. This seems to be a fair approach, because with intermediate security level we can assume that servers and slaves own access control mechanisms which prevent these attacks.

7.2.2 Requirements for SM from initialisation step

7.2.2.1 Requirements for Master

If no malicious attacker can access the master, then no SM is needed. For example, the master device has only short-range connectivity, e.g. Bluetooth, and the master is located in a physically secure place. In this document we assume that malicious attackers can access the master. Therefore SM is needed and it should provide the following security services

- Storage of SAs that are used for communication between master-slave and master-server.
- The ticket that provides authorisation domain membership must be created in the SM.

The first requirement is essential, because if some entity can get both slave and server SAs, then this entity can imitate both slave and server. If some entity can imitate both slave and server, then it can request both halves of the private key.

The second requirement is also essential, because the protected private key is used at the time when the ticket is created. Ticket creation requires the following services for the SM.

- Pseudorandom function
- Keyed hash function
- One-way function h_{dsbl}
- Computation of function h for d_1 calculation
- Either symmetric or asymmetric encryption function
- Authentication function

As a conclusion we point out that if the master is not located in a physically secure place, then it must contain a SM.

7.2.2.2 Requirements for Slave

Initially the slave contains security association SA_{Master} that is used in slave-master communications. After initialisation, the slave gets a second security association SA_{Server} from the master. This SA is used in slave-server communications.

For the slave it is not so crucial to store SAs in the SM as it is for the master. This is because if an attacker who is not a server steals SA_{Master} from the slave it must also get SA_{Master} from the server to get both halves of the key. If the server steals SA_{Master} from the slave, then the server can masquerade a slave to the master and request a new half-key. In this case server gets both half-keys and can therefore recover the whole private key.

The threat against SA_{Master} can be avoided by making the initialisation immediately after imprinting or by storing the imprinting result SA_{Master} on the SM. Therefore we recommend that if the slave does not have a smart card, initialisation for the authorisation protocol must follow immediately after imprinting.

The purpose of SA_{Server} is so that the server can verify that the slave is the correct device. However, the server will verify that the user of the slave knows the password. So even if the device or SA_{Server} is stolen, only the person who has executed the initialisation step knows the password. Therefore storage of SA_{Server} is required only for high level security.

As a conclusion the following requirements for the slave in the high level security scenario is required:

- SA_{Master} should be stored on SM.
- SA_{Server} should be stored on SM.
- Storage of random value ν.
- Integrity key for verification of encryption key is needed in SM.
- HMAC computation for integrity checking in SM
- Storage of correct HMAC value of encryption key in SM.
- Authentication algorithm in SM.

The random value ν must be stored on SM, because later the half-key d_1 will be derived from it and the password. We assume that the authentication key remains on the SM, but the encryption key may be given to the terminal. In this case integrity of the encryption key must be verified. For integrity checking the terminal provides the encryption key to the SM, which computes an HMAC value of the key and compares it to the stored HMAC value.

7.2.2.3 Requirements for Server

The server device may be located in a physically secure place and therefore it doesn't need any smart card. We assume that this is not the case. Actually it is possible that the server is a PAN component itself. Now for high-level security it is required that SA_{Master} should be stored on the SM. This requirement prevents that the slave cannot masquerade as a server to the master.

7.2.3 Additional requirements from private key operations

7.2.3.1 Requirements for Master

The master does not take a part in private key operations, so there are no additional requirements.

7.2.3.2 Requirements for Slave

In high level security it is important that the server cannot steal d_1 from the slave. Therefore

- Computation of function h for d_1 calculation is should be done in SM.
- RSA encoding method is needed in SM.

Modular exponentiation computation is needed in the SM. This is needed for computing an RSA half operation with key d_1 and in the result verification.

7.2.3.3 Requirements for Server

Assume that the server entity can be physically tampered. Now in high level security it is required that d_2 is not revealed to the terminal, because the slave device may steal it. Therefore SM in server has the following additional requirements

- Encryption and decryption algorithm
- HMAC computation
- RSA encoding method
- Modular exponentiation computation for applying half-key d_2 to message.

7.2.4 Conclusion

In this section we give a list of requirements for security functions in the SM. In Table 4 notation I means requirement for SM in intermediate security level. Notation H means requirement in high security level.

Requirement	Master	Slave	Server
Storage of SA_{Master}		H	H
Storage of SA_{Server}	I	H	H
Storage of SA_{Slave}	I		
Pseudorandom function	I		
Keyd hash function	I	H	H
Either symmetric or asymmetric encryption function	I		H
Authentication function	I	H	H
Storage of random value ν.		H	
Integrity key for verification of encryption key		H	
Storage of correct HMAC value of encryption key		H	
Computation of function h for d_1 calculation	I	H	
RSA encoding method		H	H
Modular exponentiation computation		H	H

Table 4: Requirements on the SM for delegated authorisations

7.3 Functionality split for imprinting a public key crypto system

This section considers the functionality split for imprinting a public key cryptosystem using the protocols developed by WP2. These protocols are being considered for implementation as part of the

SHAMAN demonstrator. The SM requirements on these protocols were briefly discussed in section 5.5.2.1.The following entities are involved in the imprinting protocol:

1. A PAN device known as the personal CA
2. A PAN device known as a PAN component
3. The SM which will store sensitive information (personal CA & PAN component side)

The manual authentication (MANA) protocol is used as part of imprinting. The MANA protocol relies on the user of the PAN devices to authenticate public keys and other information which are exchanged between the PAN devices. After the MANA protocol, the personal CA generates and issues a certificate on the PAN component's public key.

The MANA protocol itself does not rely on any long-term secrets to be stored on the PAN devices so it is assumed that the SM is not involved in the MANA protocol.

The functionality split for each PAN device is discussed in the following.

7.3.1 PERSONAL CA

The personal CA is designated to register or issue certificates to new PAN components that have requested to join the PAN. Clearly the private key of the personal CA is sensitive and must be protected in the TRD. We consider the following two options:

- Option 1 - Signing certificate information on the TRD
- Option 2 - Generating a certificate on the TRD

Option 1: With option 1 the TRD accepts the certificate request from the PAN component, e.g. as a PKCS#10 structure [add reference]. Once the personal CA receives the certificate request, it constructs a To Be Signed (TBS) certificate based on information contained in the certificate request and local policy information. Once the TBS certificate is created it is then sent to the TRD for signing. The TRD sends back the signed certificate to the device, the device then sends this back to the PAN component. This implies that the TRD only needs support for generating digital signatures.

Option 2: Option 2 involves constructing the TBS certificate on the TRD before signing it on the TRD. This implies that the TRD needs access to the personal CA's policy database to determine whether to issue a certificate to the requesting entity and to determine the exact contents of particular fields such as certificate validity and certificate usage. The process of constructing the TBS certificate and signing can be carried out in a single process.

The requirement to support option 1 is essential if the private key is to be protected. The security for the PAN CA is dramatically improved by keeping the PAN CA private signing key resident in the TRD at all times. This will ensure that no third party can retrieve or will have unauthorised access to the private key. It can be concluded that private key storage and private key operations must be carried out on the TRD. In this case the PAN CA device needs to support a level of integrity. It may be possible for a rogue PAN CA device to manipulate the TBS certificate before it is sent to the TRD, however, in this case even if the TBS certificate is generated on the TRD, the rogue device can manipulate the raw certificate.

Option 2 is desirable only for convenience reasons and adds no further security compared to Option 1. It might be more convenient to construct the TBS certificate on the TRD because certificate generation and signing could then be carried out as a single operation. However, constructing the certificate on the TRD will also put more processing burden on it, thus it can instead be left to the PAN device to construct and format the data. It also implies that the TRD will have to store policy information such as certificate validity and usage depending on the requesting component.

It is recommended that only private key operations and private key storage needs to be carried out by the TRD.

7.3.2 The PAN component

One option for imprinting is that individual PAN components will generate their own key pairs. In this case the PAN component should generate its key pair on the TRD to ensure that the newly generated private key will never leave the TRD. The public key that is generated can be passed to the PAN component for transfer to the personal CA for the certification process.

7.4 References

[24] Aiello, Bellovin et al.: Just Fast Keying (JFK), Internet Draft "draft-ietf-ipsec-jfk-04.txt"

8 Conclusion and outlook

With the increasing awareness of attacks on the confidentiality and integrity of mobile communication general security awareness also increases, and consequently so does the need for secure devices and components in the provision of solutions. Security modules like smart cards offer a degree of security which makes it nearly impossible, especially for an attacker with limited resources, to gain access to critical data. However, although their computational power is forecast to increase rapidly, they continue to be constrained devices, making their proper integration into secure systems a non trivial task.

With increasing processing power, smart cards open more and more fields of application and will probably also become more popular. They will also become more versatile and maybe in the future customers will have multifunction cards covering many different purposes; they will be easy to use, and they should be able to eliminate, or at least greatly reduce, the need for users to struggle with many different credentials and PIN numbers.

The WP4 work showed that handling of protocols for high degree of security, be it for network access, confidentiality or integrity protection, can be reasonably split between the mobile node itself and a hosted security module. Although the assessment of these solutions is still a demanding task and the probability of – still unknown - attacks is never zero, the gain in security is substantial. The necessary features are already available in state-of-the-art smart cards or can be built in with manageable effort.

Standardisation and adoption by the industry of new card commands, which will often be necessary for the implementation of certain protocol functionality splits, and interfaces will take a considerable time. Today, a variety of platforms and operating systems for smart cards still exist, so that a security module as described in SHAMAN can be realised in several ways. As the trend moves towards standard platforms (e.g. JAVA card), it will become probably more straightforward to define standardised reference models.

New concepts for the distribution of the workload between the security module and more powerful servers (like distributed RSA) combine security and good performance and can accelerate the introduction of secure solutions even before the smart card performance reaches levels necessary to undertake such tasks solo.

In general, the future will bring more control about security to the end user. The personal CA, for example, will enable him to have full control over a restricted number of devices; the challenge will perhaps be to make this not only secure, but also a user friendly task. In any case, in the future customers will probably also ask for more transparency of security, so they need not trust their providers and manufacturers blindly. The security module is a concept for the concentration of security in one handy and flexible device. Although the security procedures of today are still quite complex, maybe in the future - when more and more of the security functionality can be concentrated in the security module – the customer will have a much clearer idea about the security of the system that he uses.