Smart Card Based Protocol for Secure and Controlled Access of Mobile Host in IPv6 Compatible Foreign Network

Abhinav Arora R. K. Ghosh Gautam Barua

Abstract— **We present a proposal to combine the advantages of IPSec and smart cards in order to design a new protocol for secure bi-directional access of mobile hosts in a IPv6 foreign network. The protocol, called Mobile Authentication Protocol (MAP), builds a security association needed for IPsec; and uses smart cards to drive the same. An access router for the foreign network contacts an AAA (Authentication Authorization and Accounting) server in order to authenticate and authorize a mobile client hosts which approaches the it for accessing services. The access router then acts as a gateway for all subsequent service requirements of the mobile hosts. The access router interoperates between two protocols, namely, URP to communicate with clients, and AAA protocol to communicate with AAA servers. The URP has been re-christened as MAP in our proposal. It works at the application layer and uses UDP as the transport layer. Therefore, MAP works independent of the data link layer protocols. It also supports features to establish a Local Security Association (LSA) between the access router and the mobile hosts. The LSA is used to offer keying material to protect communication between a mobile host and the access router of a visited domain. The proposed design of the access router enables it to control access using IPv6 and to act as an interface between MAP and Diameter (as the AAA protocol). The network access control is secured by using IPSec by utilizing keying material offered by the LSA.**

I. Introduction

A major concern in m-commerce is security. The m-commerce solution platforms that are based on 2G networks [**?**] use a variety of methods with varying degrees of efficacy and integration to address the issue of security. With the availability of GPRS [1] and higher data rate 3G services, it is expected that IP will be available end-to-end. So most of the m-commerce traffic can then carried over IP. Thus, m-commerce will become more or less an extension of e-commerce with added benefit and features to support mobility. The security concerns and the solutions for m-commerce and e-commerce will converge, but with the constraints imposed by the limitations of mobile devices.

IP Security (IPsec) [13] is a technology that is being used in current solutions. But successful deployment of IPSec in mobile applications needs Public Key Infrastructure (PKI) for establishing Security Associations (SAs). Establishing the PKI infrastructure, Certification Authority (CA) and distribution of keys, etc., have turned out to be a difficult task. Another drawback of PKI for mobile applications is the large processing power needed for key generation and the bandwidth needed for key exchange procedures.

A related development is the adoption of smart cards as secure identification and authentication elements. Global System for Mobile Communications (GSM), which is the predominant mobile networking technology, and its evolutions uses SIM cards for authentication and authorization. Smart cards are also widely deployed in financial markets as secure, portable identification and data storage devices. An increasing number of credit cards and debit cards are incorporating smart card technology for adding security features.

The current generation of smart cards support only a single application. But with the increase in the number of applications that uses smart cards, the need for these cards to support multiple applications is growing. Emerging mobile technologies are also expected to support such multi-application smart cards. Smart cards are also being developed to support PKI, for secure storage of private keys, and also for incorporating functions like session key computation and digital signature generation.

With these technological trends in mind, it can be concluded that multi-application SIM cards will be an important part of the future mobile networking security scenario. IP, and especially IPv6, will become the predominant networking technology for mobile devices. It is expected that IPSec (which is mandatory in IPv6) will play an important role in securing m-commerce transactions.

In this chapter we present a proposal to combine the advantages of IPSec and smart cards to provide a security platform for m-commerce. A protocol for security service provider/negotiator is proposed that will be able to provide the security associations needed for IPSec. Furthermore, smart cards can be used to drive them. The smart cards may support either a single application or multiple applications. These application should be able to share credentials and keys of a single application card with, e.g., GSM or use secondary keys provided by the single application, e.g., SIM Toolkit applications.

There must be some mechanism that enables gateways to reliably identify each other. Without this, they can not sensibly trust each other and create a genuinely secure link. Any link created without some form of authentication will be vulnerable to network attacks like man-in-the-middle attacks or replay attacks. To build secure links, which exclude such attacks we use automatic keying where the two

Abhinav Arora and Gautam Barua are with the Department of CSE, IIT Guwahati, Guwahati - 781039, India. Email: {abhinav, gb}@iitg.ernet.in

R. K. Ghosh is with the Department of CSE, Indian Institute of Technology, Kanpur - 208016, India. Email: rkg@cse.iitk.ac.in

systems authenticate each other and negotiate their own secret keys. The keys are automatically refreshed periodically. In order to get access to a foreign network, the authentication of a client is done through an AAA [6] infrastructure provided by its home server. It allows the client to authenticate itself by providing the credential data depending upon the secret shared by it and the home server.

To summarize, the main focus of this work is on the design and prototype implementation of a protocol called Mobile Authentication Protocol (MAP) that allows a Mobile Host (MH) to gain access to a foreign network using a smart card as the security and authentication device. Therefore, the following issues were examined during the design and implementation of this protocol.

- Cryptographic functions.
- Applets for Java smart cards.
- AAA Architecture and AAA protocols.
- Access Router (AR) application design.
- Secure access control in the IP Layer.
- IP Security and concept of (local) security association.
- IPv6 and the stateless address autoconfiguration.
- User Registration Protocol (URP).
- Linux IPSec implementation.

The rest of the chapter is organized into nine sections. Section II provides a quick review of smart card technology. It also includes a discussion on the specific advantages, and the security features of Java smart cards which has lead us to choose the same for implementation of MAP. The AAA architecture is discussed in section III. This section also amplifies the advantage of choosing Diameter over RADIUS as the AAA protocol. Diameter supports upcoming technologies to support mobility. Section IV deals with the network security layer focusing on IPSec and related IPv6 implementation aspects over Linux. URP and MAP have been discussed in section V while the MAP overview appears in section VI. The specification of the protocol is provided in section VII; and section VIII deals with implementation. A comparison of MAP with other existing implementations of URPs is given in section IX. Section X ends with concluding remarks.

II. Smart Cards

A smart card is a credit card-size plastic card with an embedded integrated circuit. It has some memory capacity and limited computational capability. Since a card is self-contained, it is relatively more immune to security attacks as opposed to devices that have to depend upon potentially vulnerable external resources. That is why smart cards are often used in different applications which require strong security protection and authentication. The smart cards, unlike magnetic stripe cards, carry all necessary functions and information. Therefore, they do not require access to remote databases at the time of a transaction. Depending on the capability we can define the smartness of a card.

Memory Cards: A memory card can hold between 64KB to 1MB of data, but does not have a processor on the card. So, a memory card is just a read-only memory storage device. These cards have to dependent on the card reader for processing; and commonly used as pre-paid telephone cards. A memory card should not be used for storing sensitive or valuable information as the card is not used for security. Literally speaking memory cards are not "smart". Optical memory cards also belong to this category of cards. However, they provide much larger storage capacity and driven by a laser. An optical memory card can store up to 4 MB of data. It has a optical stripe which is more advanced then a magnetic stripe. Optical memory cards are used for personal identification of residents in many countries.

Microprocessor Cards: A microprocessor card, also known as a *chip card*, has greater memory storage and security of data than a traditional 125 bytes processorless magnetic stripe card. The five main components of a chip card include CPU, ROM, RAM, EEPROM and I/O controller. The card OS is stored in ROM. The RAM is used for working memory and most of the data is stored in EEPROM. The CPU is a 8-bit processor driven by a 5 MHz clock. But the trend is toward chip cards built with 32-bit RISC processors along with math co-processors. Math co-processor is needed on a card if it were to support encryption. Most cards have ROMs varying from 6 KB to 24 KB, RAM varying from 0.5 KB to 1 KB, and EEPROM varying from 1 KB to 16 KB. Four types of chip cards are used, namely, contact cards, contactless cards, hybrid cards and combi-cards. Contact cards are among the four that are commonly used. Specification of contact card standards are available under ISO 7816 series part 1-10. ISO 14443 standards define specification for contactless cards.

The major difference between a Java smart card and a conventional smart card is that the former uses Java to implement programs, whereas the latter use programs written in other languages.

The Java card specification enables Java technology [2] to run on smart cards and on the devices with limited resources. There are certain basic advantages of using a Java card instead of a non-Java smart cards.

- The applets programmed for a Java card can run on any Java-based smart card, independent of the card's vendor or manufacturer.
- Java card inherits the security features of the Java programming language. Multiple applications can coexist securely on a single smart card. Applets can be confined to operate in their respective area using applet firewall mechanism. But at the same time, to support cooperative applications on a single card a well-defined secure object sharing mechanism also exists.
- New applications can be installed securely even after a card has been issued.

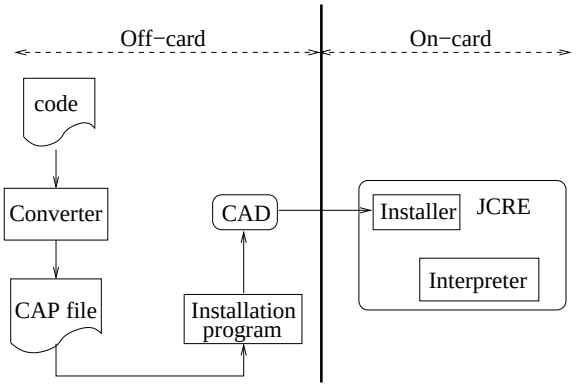

Fig. 1. Functional components of a Java card

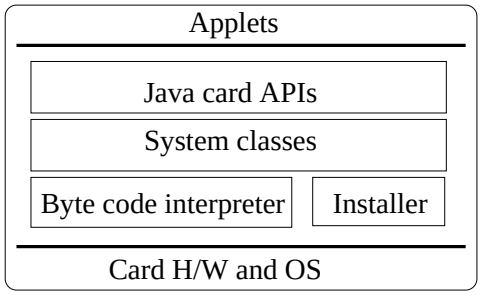

Fig. 2. Architecture of Java card

- The programs using the Java card API will run on any Java-compatible cards. In contrast non-Java smart cards can only be used in applications they are originally designed for.

A. Java Card Architecture

As shown in figure 1, the major functional component of a Java card is the Java Card Runtime Environment (JCRE). It is split into two parts: on-card and off-card. The important part of on-card JCRE is the `installer`. The `installer` works with an off-card `installation program` that transfers executable code in CAP (converted applet) file to smart card memory. The `installer` also links the applet class with other classes already installed on the card. It creates and initializes the data structures to be used internally by JCRE. Thus the job of Java Card Interpreter is restricted to execute the code in the installed CAP file. The other components of on-card JCRE are card APIs, system classes, and native methods. On-card JCRE works on the top of the smart card native system which comprises of the card OS and the h/w. The overall component level architecture is indicated by figure 2.

B. Security Features

Applets from different sources work and interoperate with each other on a Java smart card. But the private data of an applet will not be accessible by another. In fact, with Java language, it is possible to specify strict access control rules with methods and instances of a variable. Such access control rules help to secure the system by restricting

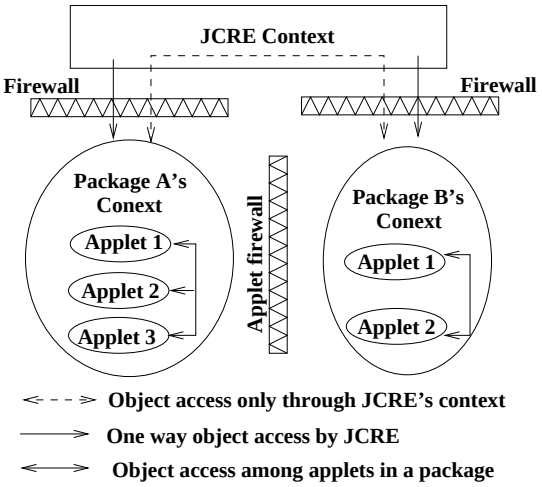

Fig. 3. Applet contexts and firewalls

the access of critical objects by untrusted codes. Java is also strongly typed. Extensive compile-time checking are done for potential type-mismatch problems. All references to methods and variables are checked to ensure that the objects are of the appropriate type. This prevents the forging of access to objects bypassing access control.

The language security policies are implemented by the Java Card Virtual Machine (JCVM). It checks every card bytecode every time it is executed. This ensures that the code is well formed; that it does not overflow or underflow the stack; and that it does not contain illegal bytecodes. The specification of Java Card API also makes it mandatory that all applets must be cryptographically signed by the card issuer. It prevents rogue applets being accepted by a card.

Since applets may contain sensitive information such a electronic currency, finger prints or private crypto keys, information sharing among applets should be carefully designed. Even a design oversight or mistake can be costly. Therefore, in Java card applets execute in isolation.

Applet isolation is achieve by a firewall mechanism. Applet firewall partitions the Java card object system in separate object spaces called *contexts*. The context is actually a group context. The applet instances of same Java package share a group context. Two applets can share objects if their respective contexts match, i.e., they belong to same Java package. The boundary between two contexts is a firewall. On the other hand, to support cooperative applications on a single card it is also necessary to allow object sharing across packages. This is achived as follows. JCRE maintains a context of its own. JCRE's context has additional privileges, namely, accesses from JCRE's context to any applet's context is allowed. So when sharing is required, context switches from one applet context to JCRE's context and then back to other applet's context. The overall picture of contexts and firewalls is provided in figure 3

JCVM ensures that a program on a smart card never escapes from the Java smart card sandbox. A Java smart

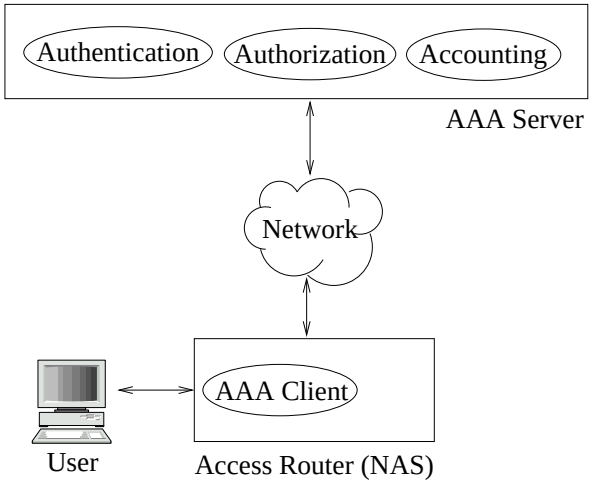

Fig. 4. AAA client communicating with AAA servers

card sandbox is a program that runs Java programs. A program running in the sandbox has to follow a number of restrictions on what it can do. For example, it can not access local file system. If a Java program tries to do anything for which it is not authorised, the sandbox terminates the program. This restricts the data available to a program and prevents one program from interfering with another.

III. AAA Architecture

Authentication Authorization Accounting (AAA) infrastructure [6], [19] is required to authenticate and authorize users for use of resources. Authentication means to verify an identity claimed by an user. Authorization is a right or a permission granted to a user to access a system resource. Accounting monitors the resource consumption data of a network for the purposes of cost allocation, auditing, and billing.

A full fledged discussion on AAA is beyond the scope of this chapter. Interested readers may refer to [**?**] for a brief tutorial on the subject. The focus our study on AAA is to realize the intended goals in the scenario we like it to operate. The protocol will need to operate in a multi-domain environment with multiple service providers as well as entities taking on other AAA roles such as a User Home Organization. Figure 4 contains a generic picture illustrating how the an end user at a network access point (NAS) communicates with AAA servers for availing AAA services. AAA servers are central repositories for storing AAA information. Sometimes multiple servers may be used for resiliency in AAA services. AAA client functions are deployed at a device which acts a entry point of the network. The device may be a terminal server, a network access router,or just another host.

In our model **AAAv** is a AAA server in the visited network and **AAAh** is a AAA server in the home network of the mobile host. The home Server (AAAh) of an access client (mobile host) has access to the AAA database which holds the authentication and authorization of that client. Whreas a visited network AAA Server (AAAv) has to proxy the mobile host's requests to its AAA home server. The most important entity in the AAA Architecture is a **Network Access Server** (NAS) or **Access Router** (AR) [5]. AR serving as the network entry point provides an interface between the IP network and the access clients. Apart from providing typical routing services, an AR has to provide the services on per-user basis. Consequently, it has to interact with a AAA Server to obtain individual client's authentication and accounting data.

Remote Authentication Dial In User Service (RADIUS) [16], and Diameter [4] are two widely used protocols which provide AAA solutions. Initially, RADIUS was designed to provide PPP authentication. IETF working group formalized the protocol in 1996. The base protocol functions and message formats are documented in RFC 2138. The RADIUS protocol can be viewed as specification for the following four functionally different classes of operations.

1. **Connectivity**: An enduser maintains connectivity with a RADIUS server via a RADIUS client that resides on the NAS and communicates over the network. A RADIUS server may also serve as a proxy for another RADIUS or an authentication server.
2. **Security**: A long term security association is maintained by using a shared secret key to secure communications between a RADIUS client and a RADIUS server. The private (secret) key is never sent over the network. Sensitive information such as user passwords contained in RADIUS messages are usually encrypted by MD5 to prevent hackers from reading them by snooping the network.
3. **Authentications**: RADIUS can support multiple authentication mechanisms including PAP (Password Authentication Protocol), CHAP (Challenge Handshake Authentication Protocol), and EAP (Extended Authentication Protocol), Unix logins or simple User/Password logins, etc.
4. **Attribute/value pairs**: RADIUS messages carry information encoded in three fields: (i) attribute type, (ii) length and (iii) value. The encoding is known as attribute/value pair (AVP). The value field contains information concerning authentication, authorization, or accounting and also for routing. Common usage of AVPs include messages carrying User-Name, User-Password, Protocol Used (such as PPP), IP address for end user, and so on. Vendors often use AVPs for certain variations in implementation of AAA schemes. RFC 2138 defines standards on the use of AVPs. But one needs to consult vendor-specific documentation for a complete lists of RADIUS attributes supported by servers and clients.

When compared with diameter, RADIUS suffers from a number of shortcomings restricting its use for roaming services, especially to mobile users. The important among these restrictions are:

Low security guarantee. RADIUS uses client/server

communication model. The communication between a RADIUS client and a RADIUS server is protected by a shared key. RADIUS provides only hop-by-hop security. So any intermediate hop can easily modify data/information without being traced. Thus, there is no end-to-end security guarantee in RADIUS. The systems based on RADIUS accounting can not be deployed on untrustworthy proxies. Encryption of only AAA data is possible in RADIUS. RADIUS can not prevent replay attack. An old packet can be replayed by a malicious NAS without being detected.

Low scalability. Only up to 255 outstanding requests can be handled by RADIUS. Allowing client mobility would mean introduction of more NASes. So the limit on outstanding message request is absolutely impractical. In addition, RADIUS has no windowing support and UDP – over which RADIUS runs – can not control flow of messages at transport layer. Therefore, RADIUS suffers from congestion problem. Consequently, RADIUS is recommended for only small to medium sized network.

Low transmission reliability. A client would never know the state of proxies or server in the chain of hops that connects it to its peer. Only timeout can recognize that some intermediate proxy is down. This causes long service disruptions. Furthermore, RADIUS silently discards the messages that do not have expected information/data. Since there is no way for a NAS to know about discarding of request, it would send same request to other server assuming that the first one to be down. The second server would also silently discard packets like the first. So the process will continue till the NAS abandons the request.

Low AVP space. Low AVP (attribute value pair) space makes it difficult to implement any local policies including transport layer security or mobile node authentication. Vendor specific commands also cannot be supported due to a limited AVP space.

Heavy processing requirement. RADIUS does not impose any alignment requirements. This places unnecessary burden on most processors. Mobile nodes are not expected to be equipped with heavy processing capabilities. This inhibits use of RADIUS for roaming services.

A. Diameter As the AAA Protocol

Diameter [4] is two times RADIUS! Not only does it include the functionalities of RADIUS, but has enhanced support for new technologies and specifically designed for roaming services. Apart from Diameter base protocol there are several extensions and applications like Mobile IP, CMS (Cryptographic Message Syntax) security, and NASREQ (NAS requirements), etc.

As opposed to RADIUS, Diameter runs over either TCP or Stream-Control-Transmission Protocol (SCTP). Servers have to support both protocols and the clients can support one of them. Both TCP and SCTP are reliable transport protocols. They provide error free, acknowledged, no duplicated transfer of packets compared to UDP. But both the protocols are heavier compared to UDP and increase the amount of traffic compared to the latter. Both SCTP and TCP support retransmission as well as windowing. Diameter protocol requires each node on the proxy chain to acknowledge each request and be responsible for retransmission of unacknowledged requests. Consequently,

1. Silent discarding of packets as seen in RADIUS is not possible.
2. The connection disruption witnessed in RADIUS by and large eliminated as unreachable nodes can be detected quickly.
3. Congestion is controlled by windowing flow of packets to servers.

Diameter uses peer-to-peer communication model. Unsolicited messages can be sent from one peer to another. This allows servers also to initiate termination of a session.

Replay attack is eliminated in Diameter due to time stamping. No packet is replied twice. So, any old packet sent by a malicious NAS can be detected. Diameter provides not only hop-by-hop security like RADIUS, but also provides end-to-end security. CMS security operates by encapsulating CMS objects in AVPs. Since Diameter has a much larger Attribute-Value Pair (AVP) space (2^{32}) compared to RADIUS (256), it is possible to use AVPs to provide CMS security. CMS security application secures messages by two main techniques (i) digital signature (along with digital certificate), and (ii) encryption. The former provides authentication, integrity and non-repudiation, whereas the latter provides confidentiality. Apart from this, local policies, if any, can also be implemented by using vendor specific commands through AVPs. Mobile IP and requirements for roaming can also be realized by leveraging the flexibility in use of AVPs. This greatly simplifies the problem of scaling up.

On the top this, Diameter insists on 32-bit alignment requirement which relieves the burden on processing, as most processor work efficiently when objects are aligned to 32-bit boundaries. Thus, all the relevant shortcomings of RADIUS with respect to security guarantee, scaling and efficiency are by and large eliminated or restricted in Diameter.

As should be obvious, Diameter is a considerably more sophisticated protocol than RADIUS; yet it is feasible to implement it within embedded devices primarily because of the improvements in processor speeds and the widespread availability of embedded IPsec implementations.

IV. Network Layer Security using IPv6

There is a need for an access router to regulate access control at the network layer, without using some data link layer protocols like PPP [3]. This is to ensure the portability of the protocol across different networks working with perhaps different data link layer protocols. Thus, an access router needs to perform source address filtering securely.

IP source address filtering is done by comparing every incoming IP packet's source address with a table of mobile host IP addresses, which have been granted access to network resources. If an incoming IP packet does meet the access requirements, it is forwarded to its desired destination. Otherwise it gets dropped and eventually an IP control message is sent back to the sender telling that access is not granted.

A. IP Source Address Filtering

IP source address filtering depends on the authenticity of the source address included in an incoming IP packet. The system is vulnerable to *IP spoofing* (i.e., creation and sending of IP packets with spoofed source addresses) attacks if there is no way of ascertaining the real sender of a received IP packet.

In our protocol, an access router not only needs to know if an incoming packet from the network node with an IP address matches the source address in the packet, but that the packet originated from an authenticated user's host. However, when dynamic host IP address configuration is used, there is no known way to relate an IP source address to a user's identity. Therefore, the access router needs a method to securely derive a user's identity from an incoming IP packet's source address. To accomplish this a protocol is needed which works at the IP layer and serves as a way to secure the integrity of at least the end-to-end immutable parts of the IP header. Our protocol works as an application layer protocol to securely bind a user's identity to the IP address of the mobile host he/she is using. The IP header integrity has to be protected with a shared secret known only to the sender and the recipient, in this case the access client and the access router. This renders IP spoofing useless since the attacker cannot generate valid IP packets with spoofed source addresses unless he/she has an access to the shared secret key.

The IP Security (IPSec) [13] protocol suite can be applied to meet the IP packet authentication requirements. Since IPSec secures network traffic in the IP Layer and integrates well with IPv6, IPSec is used in this proposed protocol as a method for protection against IP spoofing attacks.

B. IPSec

The IPSec protocol suite is a collection of security protocols that can be applied to protect network traffic in the IP layer. Beneath the essential protocols like IPSec Authentication Header (AH) [11] and IPSec Encapsulating Security Payload (ESP) [12], IPSec includes other protocol like IKE [10]. IPSec AH is a protocol used to protect the IP header against alteration by an attacker and IPSec ESP provides a method to encrypt an IP packet's payload. Additionally, IPSec describes the management and uses of IPSec Security Associations (SA).

IPSec defines two modes to process IP traffic, namely transport and tunnel mode. In transport mode, an additional security header (AH or ESP header) is added to the IP header. The tunnel mode describes an IP-in -IP encapsulation where the outer IP header contains the AH or ESP header.

In IPSec, a security association (SA) describes the security parameters of a uni-directional IP connection between two end points. IPSec SAs are stored in a Security Association Database (SADB) and are identified by a triple consisting of a destination IP address, a protocol identifier (AH or ESP), and a unique Security Parameter Index (SPI). Additionally, a SA includes the IPSec mode (transport or tunnel), the keying material used for AH or ESP, the life time of the SA, and the optional services selected within the protocol. Such an SA is used by the IPSec implementation to check the validity of incoming IPSec secured IP packets, or to insert the needed IPSec extension header into out going IP packets. The problem of using IPSec AH tunnel mode is that it can only verify the authenticity of the remote tunnel end point (by examining the outer IP header extension header), but not of the inner IP packet's sender.

Since our protocol establishes SAs as an end-to-end relation, IPSec tunnel mode is applied to secure network traffic between a mobile host(MH) and an access router (AR). The AR is an intermediate node and not the IP connection end point. The AR checks the validity of the incoming IPSec secured IP-in-IP packets and de-encapsulates them to forward to their respective destinations. On the other side, the AR has to encapsulate IP packets destined for the MH.

In the proposed protocol there is no explicit need for encryption, since AAA credentials and Local Security Association have to be secured anyway by the AAA protocol in place. Hence, IPSec AH tunnel mode is used for access control in the IP layer, and to secure the integrity of IP packets. IPSec AH makes use of an IPv6 extension header which includes a so-called Integrity Check Value (ICV) that is the result of a keyed MD5-hash function applied to end-to-end immutable parts of the IP packet Additionally, AH has a sequence number to prevent replay attacks.

The Linux operating system with kernel 2.2 or higher includes IPv6 support. Unfortunately, IPsec has not been included in the network stack. The only working implementation of IPSec in Linux is Free Secure Wide Area Network (FreeS/WAN) [9]. FreeS/WAN is open source, extends the Linux Kernel to support IPSec, and includes an SADB with an interface to user processes. For implementing our protocol using an IPSec tunnel between an MH and an AR, FreeS/WAN has been used.

The FreeS/WAN implementation has three main parts:

- **KLIPS**: Kernel IPsec (KLIPS) implements AH, ESP, and packet handling within the kernel,
- **Pluto**: Pluto (an IKE daemon) implements IKE, negotiating connections with other systems,
- **Scripts**: Various scripts provide an administrator's interface to the machinery.

In this text IPv6 version of Linux FreeS/WAN is referred to as IPSec6. IPSec6 additionally includes a Security Pol-

icy Database (SPD6) which is used to control the behaviour of KLIPS. KLIPS looks up SPD6 for every incoming and outgoing IPv6 packet. If an SPD6 entry is found whose source and destination address matches the addresses of the inspected packet, KLIPS checks if the packet conforms to the IPSec policy in place. The SPD6 therefore, acts as a powerful tool to define the security policy of an IPSec enabled host. An SPD6 entry includes the following parameters:

Type: either inbound, outbound transport, or outbound tunnel,
Addresses: the source and the destination addresses,
Tunnel address: in case the type is outbound tunnel, tunnel address defines the IPv6 address of the remote tunnel end point,
Protocol: protocol is TCP, UDP, or ANY,
Ports the source and destination Ports for protocols other than ANY are specified,
Action: defines which IPSec protocol (ESP or AH) should be used,
Algorithms: Authentication and encryption algorithms available for AH and ESP.

If KLIPS identifies an IPv6 AH or ESP header in an incoming packet, it looks up its SADB6 for a SA, which matches the packet's SPI, destination address, and IPSec protocol. In the case of AH, if a matching SA is found, the ICV included in the AH is compared with the value computed by KLIPS (if the packet is an IPSec tunnel mode packet, it gets de-encapsulated too). If ICV is not valid the packet gets dropped. Otherwise, the SPD6 is searched for an IPSec policy responsible for the respective packet(s) and its IPSec extension header(s). If such a policy is found and the packet complies with it, the packet gets forwarded.

When an IPv6 packet gets sent, KLIPS first consults its SPD6 to see if there is a security policy which required IPSec processing for this packet. If a policy is found and there is a matching SA in SADB6, the required extra header is included in the IPv6 header. In the case of ESP the packet payload also gets encrypted. Furthermore, if the policy prescribes tunnel mode, the packet gets encapsulated too.

V. User Registration Protocol

A User Registration Protocol (URP) [14] allows a user to register in the network by providing his/her identity and authentication information to the local network, which validates the user, charges him/her, and authorizes the use of resources with the help of an AAA infrastructure.

A. Designing a URP

Most existing protocols operate at the data link layer which make them usable only for one specific access technology. We, therefore, designed a URP which is independent from the access network's data link layer protocols. Interoperation with AAA protocols like RADIUS or Diameter is a must for a URP. Since an access router – which is an end point for URP – communicates directly with AAA servers, there is no need for a URP to include AAA protocol functionality This interoperation requirement, to some extent, prescribes which authentication credentials and authorization data an URP must transport between a mobile host and an access router. As no point-to-point data link layer protocol information available at the application layer, there has to be a way for an access client (MH) to discover the location of an access router.

With IPv6, the best method for discovery of access router (AR) is to extend the use of the IPv6 stateless address auto-configuration feature. The AR location is added to router advertisement messages. An URP should also have a method to establish a Local Security Association (LSA) [8] between an AR and an MH. Additionally, it should support other features like performing LSA re-negotiation in case the lifetime of an LSA has expired. Since an LSA is established with the help of an MH and its AAA home server, re-negotiation of an LSA does not depend only on a URP. Also, URP communications has to be secure otherwise an attacker could gain access to network resources that he/she has no permission for. In this context, secure communication means authenticated and replay protected communication. There is no explicit need for encryption since AAA credentials and LSA keys have to be secured by the AAA protocol anyway.

B. Local Security Association

SAs can be further defined as either inter-domain SAs or Local SAs (LSA). An inter-domain SA is established between entities belonging to different network domains, whereas an LSA is used between entities that are located in the same network domain. Another difference is that an LSA typically has a shorter lifetime compared to an inter-domain SA. In [8] the concept of a user specific LSA called Temporary Shared Key (TSK) has been proposed. A TSK is established between an access client (MH) and the visited domain's AR with the involvement of the AAAh (Home AAA server of MH). Our proposal is to establish a TSK between a mobile host and an access router to secure the Network Layer access control. Once established, the TSK is used as the IPSec SA to build an IPSec tunnel between the MH and the AR.

To build a TSK, a AAA home server AAAh has to generate a unique security key and then distribute it to the AR and the MH. Since the MH does not interface directly with the AAAh, the key has to be sent only through the AR. The key distribution has to be made as secure and as reliable as possible because if the TSK is lost the whole session will be insecure.

Figure 5 shows the pre-established SAs in a typical AAA environment. The TSK has to be sent from the AAAh to the AR in a secure manner. To accomplish this, the pre-established SAs **SAvh** and **SAmh** are used to securely transfer the TSK up to the AR. If the AR receives the TSK parameters, it has to send these parameters securely to the MH. But since the AR does not share a pre-established SA

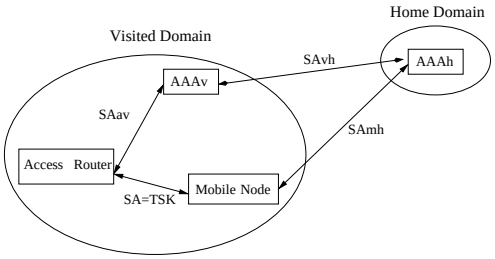

Fig. 5. Security associations in a AAA environment

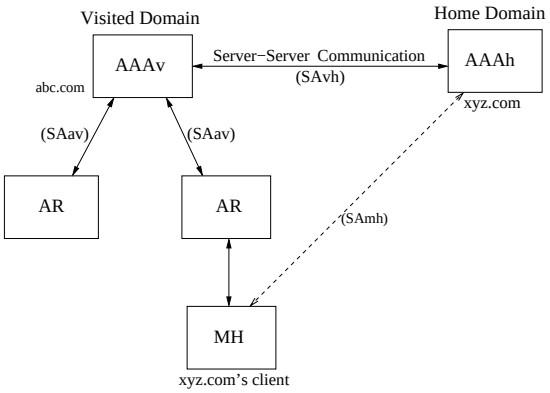

Fig. 6. Logical entities of the model

with the MH at this stage, the AR needs some more information from the AAAh server to securely distribute the TSK parameters. This extra information consists of either the SAmh parameters, or the encrypted TSK parameters using SAmh. In the latter case, the AR has to only forward the received SAmh encrypted TSK parameters to the MH. We have used this approach in developing the protocol, since it is more secure as the AR does not need to know the SA parameters (including the long term key shared between an MH and its AAAh Server). After the MH has received the TSK parameters, the TSK between the MH and the AR is established as shown in figure 5

The LSA key is encrypted by using 3-DES [15] with a 192-bit key. The key is generated by recursively applying an HMAC-MD5 [17] hash function on the long term shared key and by interlacing the so computed MD5 digest.

VI. Protocol Overview

The aim of our protocol is to authenticate a mobile host roaming in a foreign network to its home server so that it can access the network resources of the foreign network and then develop a secure link between the mobile host and the access router. Further exchanges of messages between the two can then be safe. The protocol ensures the safety of messages from various attacks like man in the middle attack, replay attack and IP spoofing. The important issue that needs to be addressed by our protocol is that there should be as few message transfers as possible between a mobile host and its home server. This is to tackle the constraints which naturally arise due to mobility and the wireless communication infrastructure. Our protocol is also designed to suit the limited processing capability of smart cards (Java card). With the resources available to a smart card, the complexity and the size of cryptographic algorithms are restricted to the extent possible.

A. Mobile Authentication Protocol

The URP implementation, called MAP, discussed in this section was developed covering all URP design specification requirements as discussed earler in subsection V-A. MAP is an application layer protocol. It inherits most of its features from EAPoUDP [7] (a variation of the Extensible Authentication Protocol [3]), and operates on UDP in the transport layer. There are other possible choices for the transport layer protocol like, TCP or ICMP. UDP was chosen for MAP because it is the most generic delivery method among those mentioned above. However, UDP is not reliable. Therefore, it introduces the need to implement retransmission and acknowledgement strategies, as it would be the case when using ICMPv6. But we do not require full TCP functionality which introduces many overheads. Considering a MH's processing power and bandwidth, it will be better to work with a light weight protocol.

The AAA protocol inter-operation requirement is met by using EAPoUDP parameters as AAA credentials. EAPoUDP assumes that prior to authentication the MH has configured a valid IPv6 address for itself and received the AR location during IPv6 stateless address autoconfiguration [18]. MAP messages are secured where needed with a challenge option and `AuthData` which is the result of a HMAC-MD5 one-way hash function applied to the end-to-end immutable parts of the MAP message. By doing so, *man in the middle attack* and *replay attacks* are eliminated.

After a successful authentication, an MH gains access to the foreign network only for a limited time period. After the allocated time period the AAAh server sends a TSK Update to the AR and the MH by providing required TSK parameters to re-establish the LSA between the MH and the AR.

B. The Logical Entities

The five main logical entities of the protocol and their mutual interactions have been depicted in figure 6.

MH denotes a mobile host visiting a foreign network. The access router AR allows the MH to register and be authenticated by the network. AAAv is the AAA server of the visited network; and AAAh is the AAA server in the home network of MH. The end result of the authentication procedure is that a Temporary Shared Key (TSK) is set up between MH and some agent (an acccess router) in the visited network.

C. Security Elements

The security elements deployed by the proposed protocol include the security association and cryptographic algorithms. These combine together to ensure secured bidirectional access between a MH and a foreign network.

C.1 Security associations

The security associations are put in place by the following elements and guiding principles.

SAvh: MAP assumes that the AAAh and the AAAv share a long term SA, SAvh, and that it is not specific to any particular user.

SAav: It is assumed that each network has its own security mechanism and an SA, SAav, allows the entities in the same network to communicate in a secure and mutually authenticated way.

SAmh: It is assumed that each user, as a part of a subscription agreement with a home domain, acquires a long-term security association (SAmh) with her/his home domain. In fact, Mobile IPv6 mandates the existence of a security association between an MH and its AAAh.

TSK: For a LSA to be adopted between the user and the visited domain, the user and the visited domain must have a set of common security algorithms that can be used to support the LSA.

C.2 Cryptographic consideration

There is no negotiation of cryptographic algorithms in our protocol. All the algorithms are pre-decided. We use 3DES [15] in CBC mode for encryption and HMAC-MD5 [17] for authentication. The 3DES algorithm is used for generating the keys (like TSK between MH and AR). The HMACS-MD5 [17] is used as a one way hash function. The 3DES algorithm uses a 192 bit key. We use the EEE approach, that is, the encryption algorithm is used three times, applying different keys each time. The HMACS MD5 algorithm uses a 128 bit key.

D. Security Features

As discussed earlier, the basic security features that has to be supported by MAP are: authentication and authorization, and establishing a LSA between MH and AR. When these features are guaranteed, the protocol sets up a IPSec tunnel between the access router and the mobile host.

D.1 Authentication/authorization

Authentication is required before providing network access to the mobile user. Not only does the user needs to be authenticated, the network providing the service should also be authenticated. So, the authentication mechanism which we think would be appropriate for two way authentication is to have the network broadcast a Local Challenge over the access link, for example along with the Router Advertisement messages.

D.2 Setting up a local security association

We can improve the protocol by setting up a LSA between a user and the visited network when the user is roaming. The adoption of an LSA allows for optimizations and empowers the visited service provider to authenticate the user at any time and perform key distribution without the involvement of the home domain.

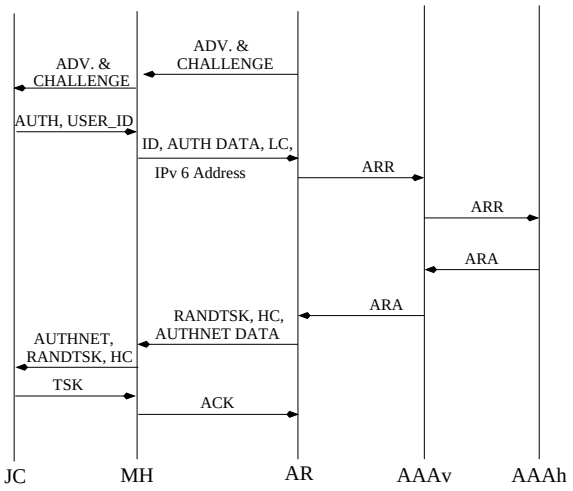

Fig. 7. Protocol Specification

Without the use of an LSA, user authentication, network authentication, key distribution, etc., between the user and agents in the visited domain is usually based on the long-term SA between the user and its AAAh. When TSK is adopted, the user receives a notification that TSK is to be used. Therefore, the user would use SAav of the visited network instead of AAAh. It saves the round trip times between the visited and home networks, thus reducing time delay and the network load.

D.3 Building an IPSec tunnel

The protocol uses the LSA formed between a user and an access router of the visited network to form an IPSec tunnel between them, thereby securing any exchange of messages between the two.

VII. Protocol Specification

Figure 7 provides the important time-lined actions of entities for the Java card driven authentication of MHs in foreign networks.

When the user inserts the smart card in the card acceptance device, it prompts the user to enter a Personal Identification Number (PIN). If the user enters the correct PIN then the client application running on the MH gets activated. When an MH enters a new subnet, it receives a Router Advertisement with a Local Challenge as shown. This router advertisement is broadcast periodically by an Access Router (AR).

A. Request from MH

After receiving a router advertisement from an AR in the visited network, the MH sends a message to the AR seeking the services to be delivered to the access router. MH constructs a tentative IP Address, called *Care of Address (CoA)*, and replies with an EAP Response message [7] with the following parameters:

1. **Local Challenge(LC)**: This is the value sent by the AR in the Router Advertisement. The LC is a *random string*. Its purpose is to ensure freshness of the messages exchanged so as to avoid replay attacks.
2. **Client Identifier (user_id)** consisting of the client's user_id (provided by the smart card). This contains information of the client's home network also.
3. **VN_ID** is the id of the visited network and is sent in the router advertisement from where it is copied.
4. **AAA Credential Option** is sent by the smart card. It is constructed by concatenating all of the preceding parameters and the long term shared key between the MH and its AAAh (SAmh) and applying the algorithm agreed upon, which in our case is HMAC-MD5:
   `AUTH = HMAC-MD5(LC,user_id,VN_ID,SAmh)`

B. AR's response to MH's request

The access router (AR) first verifies the freshness of the request thanks to local challenge (LC) and then performs duplicate address detection on the care-of-address. If it fails, the AR replies to the MH with the code "ADDRESS_IN_USE". Otherwise, it creates a Diameter ARR (AAA-Registration-Request) message carrying the following information to the AAAv (containing attribute-value pairs (AVPs)):

1. **User Name AVP**: to carry the client identifier (user_id).
2. **Challenge AVP**: to carry LC for replay attack protection.
3. **EAP AVP**: to carry the authentication data `AUTH`, for mutual authentication.
4. **VN_ID**: the visited Network's Id as received from the MH
5. **care-of IP address**: received from the MH (MH_Ipaddr).

C. Actions of AAA server of the visited domain

When an AAAv receives an AAA registration request message, ARR, it verifies the message is coming from a valid AR, and forwards the message to the MH's AAAh by looking into MH's client identifier.

D. Actions of AAA home server

When AAAh receives an ARR message from an AAAv, it first verifies whether the message originated from a valid AAAv, then from the information contained in several AVPs it executes the following procedure.

1. Extracts `LC, user_id, AUTH, VN_ID`, and `MH_Ipaddr` and authenticates the user using the client identifier provided by the MH as the MH's identity, the `AUTH` information sent, and the long term shared key between the two (SAmh). That is, it checks if `HMAC-MD5(LC, user_id, VN_ID, SmAh)` is equal to `AUTH`.
2. Stores `MH_Ipaddr` for future use.
3. Generates a random number `HC` to ensure freshness of a message.
4. Creates an `AUTHNET` value as follows to ensure freshness of the message to be sent:
   `AUTHNET=HMAC-MD5(HC,user_id, VN_ID,SAmh);`
5. Creates a random number `RANDTSK` which is used as the basis for the session key to be defined. `RANDTSK` is sent encrypted as `TSK` as follows: `TSK = 3DES(RANDTSK, SAmh)`
6. Finally sends a Diameter ARA (AAA Registration-Acceptance) message as reply to the AAAv and this contains `RANDTSK, HC, TSK, VN_ID, user_id`.

E. Response to ARA message by AAA server of visited domain

When AAAv receives an ARA message from AAAh, it forwards it to the AR that sent the original request.

E.1 AR's response ARA forwarded by AR

In response to an ARA message from AAAv, the AR converts the message to the EAP format and sends it to the MH; this message carries:

1. the authentication data, `AUTHNET`
2. the random number `HC`
3. the key generation number `RANDTSK`

E.2 MH's response to AR's message

1. The MH sends the information to the Java smart for verification.
2. The Java card calculates `AUTH = HMAC-MD5(HC, user_id,VN_ID, SAmh)` and compares it with `AUTHNET` to authenticate the information. Then it calculates `TSK` from `RANDTSK` as outlined above. This is returned to the MH.
3. MH now uses `TSK` to build an IPSec tunnel between itself and the AR and the link is established.

F. TSK Update

The TSK that is established between the MH and the AR usually has a limited lifetime. When this expires, the AR requests a new `TSK` from the AAAh via the AAAv. The actions involved in such a `TSK` update are illustrated by figure 8. The basic procedure is similar to what has already been described and it will not elaborated further.
`TSK = 3DES (RANDTSK, SAmh)`

- AAAh sends RANDTSK-AVP and TSK-AVP carrying `RANDTSK` and `TSK` to the visited network server.
- It waits for the report from the visited network server.

VIII. Implementation Details

The implementation of the proposed protocol was done by setting up an IPv6 test bed to drive the IPSec. The topology was configured as indicated by figure 9.

FreeS/WAN [9] with IPv6 support (IPSec6) has to be installed in order to use **MobileApplication** and **AccessRouter** applications. As mentioned earlier FreeS/WAN is a Linux implementation of the IPSec protocols.

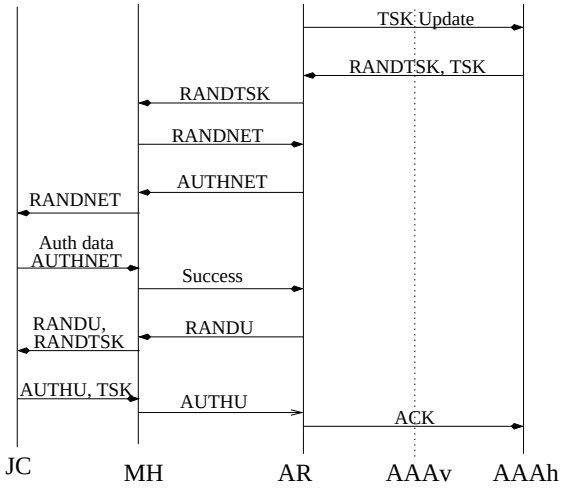

Fig. 8. TSK Update

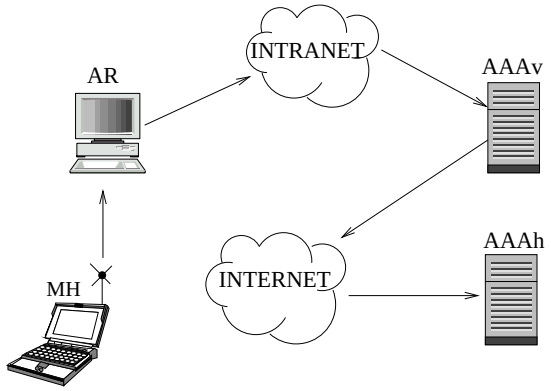

Fig. 9. Testbed configuration

Since IPSec operates at the network layer, it is flexible and can be used to secure nearly any type of Internet traffic. Two applications, however, are extremely widespread. These are

Virtual Private Network (VPN). It allows multiple sites to communicate securely over an insecure Internet by encrypting all communication between the sites.

Road Warriors. It connects the office from home, or perhaps from a hotel somewhere.

In our protocol, we used the Road Warrior implementation, changing it to suit our requirements. The implementation of the protocol involves the software entities described below.

A. Java Card Simulation

Whenever the smart card is inserted in the card reader, the client application prompts the user to produce a PIN (Personal Identification Number). It is checked with the PIN stored in the EEPROM of the smart card for user authentication. If the user is not authenticated then no further transactions can be done.

There will also be a method **Admin(byte[ ] password)** which will be used for the smart card issuer to initialize or

Fig. 10. Smart card interface

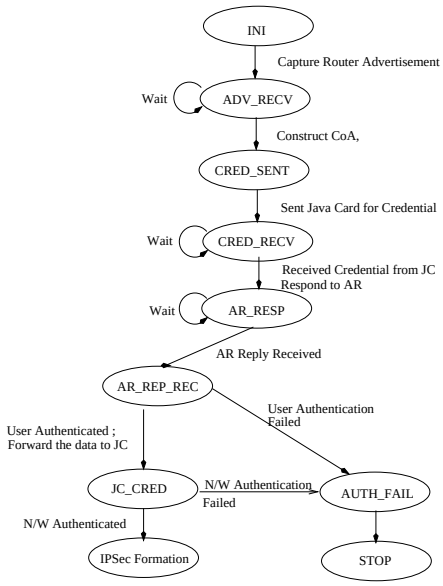

Fig. 11. Mobile Application State Machine

change the crypto keys, choose the crypto algorithms, set the user ID, set the PIN, and other initialization parameters for the smart card applet. The mobile host will have an application running which will use the interface provided by the smart card to communicate with the applet inside the smart card. This process is depicted in figure 10

B. Mobile Application Software

A program called **MobileApplication** implements the client side of MAP and has an interface to SADB6 of IPSec6. If a mobile user wants to get access to the Intranet, he/she will run this program with the **Java Card** attached to his/her machine. **MobileApplication** establishes an AH tunnel mode connection between the system and an Access Router. The application is written as a single threaded program in Java which has an interface both with the Java Card and the AR.

B.1 Protocol state machine of MobileApplication

Figure 11 shows the protocol state machine implemented in **MobileApplication**. It essentially implements the protocol described above.

After **MobileApplication** has captured a Router Advertisement from the AR, it sends the received information to the Java Card for credential evaluation and by that time, it configures its own *Care of Address* (CoA) using the visited network domain address. It forwards the authenticated data to AR with its own calculated IP address.

If the MH receives a positive MAP Response from the AR, it forwards the intended information to the Java Card,

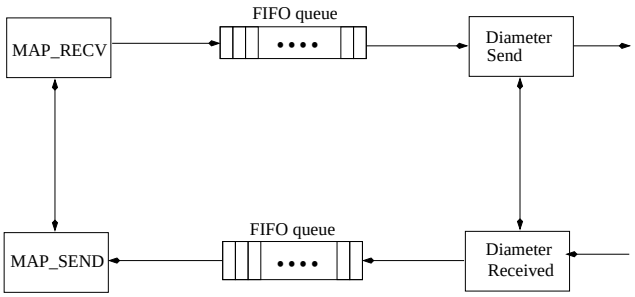

Fig. 12. Access Router architecture

to authenticate the network. After the network authentication becomes successful, it installs the needed IPSec SAs in its SADB6 using the TSK included in the MAP response.

C. Delay Server

This application is written mainly to simulate real network conditions. It inserts virtual hops between any two different applications. This application is started by the source and waits to receive data from it, then it inserts the required number of hops by sending data again and again to the local port before forwarding it to the destination. It provides an interface by which the number of hops to be inserted between the applications can be changed.

D. Access Router Application

AccessRouter is a prototype implementation of an Network Access Server (NAS) acting as a Policy Enforcement Point (PEP). **AccessRouter** application consists of four different concurrent threads. **AccessRouter** implements MAP as well as an extended version of Diameter (Diameter with TSK support). Additionally, it also controls the behaviour of IPSec6 with proper interface to the kernel space.

D.1 Access Router Architecture

An AccessRouter process consists of four concurrently executing threads as shown in the figure 12

The **AccessRouterMAP** threads are responsible for the communication with an MH using MAP while the **DiameterSend** and **DiameterReceive** threads implement the communication with the AAA Server using Diameter. The **dia_out_queue** and the **map_out_queue** are both FIFO protocol messages queues used to send messages in an asynchronous manner.

Figure 13 shows the protocol state machine of **AccessRouter**. In the initial state of AR, it starts sending a Router Advertisement periodically to the network. As soon as a packet is received by **AccessRouterEAP** or **DiameterReceive**, and if the protocol message included in the packet complies with the protocol state machine, a state change including the respective action as shown in Figure 13 is executed. Otherwise, the packet gets silently discarded.

During MH authentication, **AccessRouter** has to establish the IPSec AH tunnel mode SAs with the help of

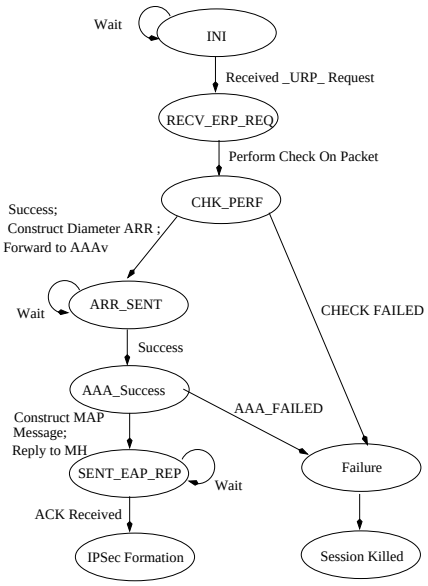

Fig. 13. Access Router State Machine

the TSK parameters received from the AAA Server. Additionally, two entries have to be added in the database, one for inbound (Network to MH) and one for outbound (MH to Network) network traffic. The needed IPSec SAs have to be added to the SADB6. Since MAP runs over unreliable UDP as a Transport Layer protocol, it introduces the problem that an AR does not know the protocol state of an MH without the reception of an acknowledgement message.

An IPsec SA needs a unique Security Parameter Index (SPI), which is a 32 bit number identifying an SA. An SA's SPI has to be same in both end systems of an IPSec connection. It is therefore necessary for the **AccessRouter** to send the SPI to the MH after the successful addition of the SAs to its SADB6 (this was not shown in the protocol description). The base SPI is the SPI of the first added SA and is used since an IPSec AH tunnel mode connection needs more than one SA to be added to the SADB6. FreeS/WAN has reserved SPI numbers in the range from **0x100** to **0x1000** for manual keying, which describes the establishment of SA without IKE which is in compliance with our protocol. Therefore, **AccessRouter** uses SPI values between **0x200** and **0xfff** in order not to interfere with a running IKE instance and to still allow manual keying for our purpose.

E. User-Kernel Space Interface

There is a need to implement user-kernel space interfaces for the SADB6 and SPD6 from IPSec6. Since the aim of our implementation is just to set up a secure connection between an MH and its home server by setting up an IPSec tunnel between the MH and an AR, user-kernel space interface can be further extended to allow proper and restricted movement of traffic between MH and AR.

Linux offers a virtual filesystem called `/proc` that enables the communication between user space processes and

kernel space processes. A kernel which offers such a `/proc` file has to implement a read and a write file handler function for this file. If a user process reads or writes to a `/proc` file, the respective read or write handler function of a kernel process is called. A new IPSec6 policy can be added/deleted to the SPD6 by appropriate editing of the virtual file `/proc/net/spd6`.

It is necessary for the application to keep track of the SPD6 rule numbers, since these numbers do possibly change after rule deletion.

A user process communicates with the SADB6 by using a PF_KEY socket. FreeS/WAN does implement such a socket type for the Linux Operating System. PF_KEY supports only the `write()` and `read()` BSD socket routines.

F. Visited Network Server Application

This application implements the visited network AAA Server (AAAv). It receives packets from an AR and performs checks to determine if the packet has come from a reliable AR. If not then the packet is dropped with a failure packet sent to the source AR. If the packet passes the reliability check then it is parsed to find if it is a local request or it has to be forwarded to a server of some other domain with which it has some pre-shared contract. After the request for authentication is through then it may optionally store data for accounting purposes.

G. Home Server Application

This application performs as a home to its mobile host (AAAh). It has the long term key stored in an internal database for every mobile host (the SAmh's). If a request for authentication comes from some foreign domain, it parses the AVPs of the packet to get the credential to authenticate the user.

If authentication succeeds it calculates some other credentials to authenticate itself to the MH and also generates a TSK to be shared between MH and AR (as described in the protocol description). `HomeServer` application also maintains some `timeout` for each TSK provided to MH so that with elapsed `timeout` it can initiate the TSK update function (An AR can also initiate this timeout).

IX. Experimenting with the MAP Implementation

Our protocol essentially combines the advantages of symmetric key cryptography and the use of a temporary shared key. The implementation of the protocol, however, was mainly focused to investigate its feasibility. But the implementation provides enough insight to be able to evaluate the MAP against other secured controlled access methods of a MH in a foreign network.

There are two other known protocols which may be deployed for secure access of MHs in a foreign network. These are PKI and IKE. An evaluation of merits of the proposed protocol is done best by comparing it against the two existing protocols.

A. PKI versus MAP

PKI uses asymmetric key cryptography. It uses two separate keys One of which is public, and the other is private. But in a PKI based authentication system MH and its home server would have to share the private key. The main drawback of this scheme is that after some time, an attacker can probably break the private key by eavesdropping the communication for some time. Another limitation of PKI is the problem of communicating the secret keys between the two entities, and if the key is somehow lost then communication can not take place. These shortcomings have been taken care of in MAP by refreshing the keys between MH and AR periodically and securely under the supervision of the home server. Also in MAP we are using smart cards as the medium for transporting the keys where these are hardwired and cannot be read by any malicious user. Thus both the above limitations of PKI are eliminated. Apart from these limitations, establishing the PKI infrastructure of a Certification Authority (CA) for the distribution of public keys has turned out to be a difficult task both economically and practically.

B. IKE versus MAP

The default IPSec key management protocol is IKE. IKE works in two phases. In the first phase, IKE can work with either main mode or aggressive mode where it negotiates the IKE Security Associations. The main mode requires 6 messages exchanges as compared to 3 messages exchanges by the aggressive mode. In the second phase SA's formed are used to provide authentication, secrecy and data integrity which again takes 3 more messages. Hence in all, forming the SA using IKE requires 6 or 9 messages. Also, IKE runs on the infrastructure of PKI which makes it computationally more expensive.

On the other hand, the process of establishing a TSK takes only 3 messages and this forms an SA between two entities with the same level of security. TSK sharing gives the serving system significant load control over the authentication and key distribution of a visiting MH: the key refreshing and new key distribution procedure can be based on this temporary shared key stored in the AR thus saving round trips with the home network.

X. Conclusions and Future Work

This chapter has described a protocol to provide secure and authenticated access to roaming mobile hosts in a foreign network using a Java smart card. The advantages of using a smart card to store security information and to implement the basic authentication functions have been exploited in this protocol. It has been designed so that it performs better in comparison to the existing protocols. Limited computing power, limited battery power, and possibly limited bandwidth in a mobile host have been taken into account while designing the protocol. It provides for the delegation of part of the authentication function to an access router of the foreign network. This helps reduce traffic to and from the home network. Further, the proto-

col uses temporary shared keys to limit the damages due to the break of a key.

Acknowledgements

The implementation of the protocol proposal was carried out by the first name auther and Deepak K. Singh at IIT Guwahati as a requirement for their major undergraduate project. The authors acknowledge the contribution of Deepak K. Singh.

References

[1] General packet radio service. http://www.gsmworld.com/technology/gprs/intro.shtml.

[2] Javatm technology. http://www.java.sun.com.

[3] L. Blunk and J. Vollbrecht. Ppp extensible authentication protocol (EAP), RFC 2284. http://www.ietf.org/rfc/rfc2284.txt, Mar. 1998.

[4] P. R. Calhoun, H. Akhtar, J. Arkko, E. Guttman, A. C. Rubens, and G. Zorn. Diameter base protocol,<draft-ietf-aaa-diameter-09.txt>. Internet-Draft <draft-ietf-aaa-diameter-08.txt>, Nov. 2001.

[5] M. B. D. Mitton. Network access server requirements next generation nas model, RFC 2881. http://www.ietf.org/rfc/rfc2881.txt, July 2000.

[6] C. de Laat, G. Gross, L. Gommans, J. Vollbrecht, and D. Spence. Generic AAA Architecture, RFC 2903. http://www.ietf.org/rfc/rfc2903.txt, Aug. 2000.

[7] P. Engelstad. EAP over UDP, internet-draft <draft-engelstad-pana-eap-over-udp-00.txt>. Internet-Draft <draft-engelstad-pana-eap-over-udp-00.txt>, Feb. 2002.

[8] S. M. Faccin and F. Le. AAA local security association (LSA): The temporary shared key (tsk), <draft-le-pana-lsa-tsk-00.txt>. Internet-Draft <draft-le-aaa-lsa-tsk-00.txt>, Jan. 2002.

[9] J. Gilmore, H. Spencer, R. G. Briggs, and H. Redelmeier. Frees/wan project. http://www.freeswan.org.

[10] D. Harkins and D. Carrel. The internet key exchange (IKE), RFC 2409. http://www.ietf.org/rfc/rfc2409.txt, Nov. 1998.

[11] S. Kent and R. Atkinson. Ip authentication header, RFC 2402 (standard). http://www.ietf.org/rfc/rfc2402.txt, Nov. 1998.

[12] S. Kent and R. Atkinson. Ip encasulating security payload, RFC 2406 (standard). http://www.ietf.org/rfc/rfc2406.txt, Nov. 1998.

[13] S. Kent and R. Atkinson. Security architecture for the internet protocol, RFC 2401. http://www.ietf.org/rfc/rfc2401.txt, Nov. 1998.

[14] Y. Ohba, J. Kempf, P. Robert, B. Subbiah, B. Patil, H. Haverinen, and H. Soliman. Usage scenario of a user registration protocol (URP), internet-draft <draft-ohba-urp-usage-scenarios-00.txt>. Internet-Draft <draft-ohba-urp-usage-scenarios-00.txt>, Dec. 2001.

[15] P. M. P. Karn and W. Simpson. The esp triple des transform, RFC 1851. http://www.ietf.org/rfc/rfc1851.txt, Sept. 1995.

[16] C. Rigney, S. Willens, A. Rubens, and W. Simpson. Remote authentication dial in user service (RADIUS), RFC 2865. http://www.ietf.org/rfc/rfc2865.txt, June 2000.

[17] R. Rivest. The md5 message-digest algorithm, RFC 1321. http://www.ietf.org/rfc/rfc1321.txt, Apr. 1992.

[18] S. Thomson and T. Narten. IPv6 stateless address autoconfiguration, RFC 1971. http://www.ietf.org/rfc/rfc1971.txt, Dec. 1998.

[19] J. Vollbrecht, P. Calhoun, S. Farrell, L. Gommans, G. Gross, B. de Bruijn, C. de Laat, M. Holderege, and D. Spence. AAA authorization framework, RFC 2904. http://www.ietf.org/rfc/rfc2904.txt, Aug. 2000.