NAAP: A User-To-Network Authentication Protocol

Henry Haverinen
Nokia Mobile Phones

Abstract This paper presents the design of an IP-level user-to-network authentication protocol called the Network Access Authentication Protocol (NAAP). The protocol is an extensible UDP-based protocol that is used by a mobile terminal and an edge router for IP network access authorization for example on public Wireless Local Area Networks (WLANs). NAAP can use any Extensible Authentication Protocol (EAP) method that is capable of distributing a session key. It includes authentication service discovery, message integrity protection, protection against message replays, deregistration and a mechanism to detect if a terminal has silently left the access network. The protocol has been implemented and it is used in the Nokia Operator Wireless LAN solution [1] with GSM SIM authentication.

Keywords: Network access authentication, user-to-network authentication, extensible authentication protocol

1. INTRODUCTION

Network access authentication is often performed at layer 2 between a terminal and the network. For example the Point-to-Point Protocol (PPP) [2] performs network access authentication before the Internet Protocol (IP) configuration phase. Data services in current cellular networks also implement network access authentication below the network layer. IEEE is specifying new authentication procedures for IEEE link layers such as IEEE 802.11 WLANs. These protocols are sometimes called user-to-network protocols, because a user device and a network element performing attendant functionality use the protocol.

Usually, the attendant uses another protocol to communicate with backend servers in order to authenticate the user, authorize the user for network access and account for the usage of resources. Examples of such

The original version of this chapter was revised: The copyright line was incorrect. This has been corrected. The Erratum to this chapter is available at DOI: 10.1007/978-0-387-35584-9_19

O. Martikainen et al. (eds.), *Smart Networks*

network-to-network Authentication Authorization and Accounting (AAA) protocols are Radius [3 - 5] and its successor Diameter [6], used in IP access networks with PPP.

Link-layer user-to-network protocols are currently not available for all access technologies such as the current Ethernet and Wireless LAN implementations. Implementing such link-layer protocols in an existing system is hard because it implies changes to low level software or even hardware. Because link-layer protocols are specific to a certain link layer, the functionality needs to be implemented separately for each link layer.

Mobile IPv4, specified in [7 - 12], includes access network authentication features implemented at the network layer. The visited network may deploy foreign agents, with which mobile nodes must register in order to get network access. The foreign agent may make use of an AAA protocol to authenticate and authorize the mobile node. However, it does not make sense to deploy Mobile IP just for network access authentication on hosts that do not need IP level mobility. Mobile IPv4 implementations are non-trivial to deploy on current operating systems. Mobile IP also requires the network to have home agents, and to use IP tunneling, which are unnecessary if the goal is to provide for network access authentication only.

The Network Access Authentication Protocol (NAAP) is a user-to-network client/server protocol, which performs authentication and access authorization over IP and hence decouples authentication from the access technology and the link layer. NAAP can be used over any link layer, including WLANs, Ethernet, ADSL etc. *Figure 18* shows the architecture of the authentication network. NAAP protocol entities, the NAAP Client and the NAAP Agent, reside in the user terminal and the edge router, respectively. The edge router communicates with the user's AAA server using a network-to-network AAA protocol. Although the NAAP Agent usually resides in the edge router, it could be placed deeper on the network as well.

The NAAP Client software can easily be implemented as add-on software on current operating systems without changing network cards or drivers. It is considerably easier to implement and deploy than Mobile IP client software, which typically requires system level programming. A NAAP Agent is implemented on a router, where NAAP can be used in conjunction with a network-to-network AAA protocol. Because access enforcement (packet filtering) is implemented at the network layer with access control lists, NAAP allows the access network to provide free local services that are available without any authentication. For example a hotel network may allow free access to the hotel Intranet but require authentication for global Internet access.

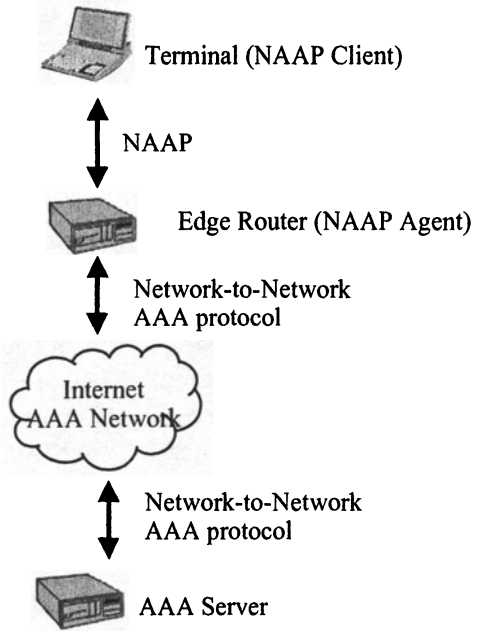

Figure 18. Authentication network

When a terminal enters an access network, it obtains an IP address for example with the Dynamic Host Configuration Protocol (DHCP). NAAP is independent from the method of obtaining the IP address. A terminal visiting a NAAP-enabled access network must register with the local NAAP Agent in order to get network access. After obtaining the address, the terminal uses the NAAP Agent discovery functionality to detect if the local network requires NAAP registration and to learn the IP address of the local NAAP Agent. The terminal then registers with the NAAP Agent and is finally granted network access.

Many NAAP features are modeled after Mobile IPv4. Agent discovery, protocol message formats, message integrity protection and replay protection resemble the equivalents in Mobile IP.

NAAP re-uses the PPP Extensible Authentication Protocol (EAP), discussed in Section 2 of this paper, in a novel way. Section 3 shows an overview on the NAAP authentication procedure. Section 4 discusses local NAAP messaging between the NAAP Client and the NAAP Agent. NAAP security considerations are covered in Section 5, and the use of NAAP with Virtual Private Network software in Section 6. Finally, Section 7 contains conclusions.

2. EXTENSIBLE AUTHENTICATION PROTOCOL

In first versions of the Point-to-Point Protocol an implementation that wanted its peer to authenticate with some specific authentication protocol requested the use of that authentication protocol during link establishment phase. The PPP implementations in both ends of the connection were required to know the authentication protocol. If a new authentication protocol was designed, existing Network Access Servers (NAS's) had to be updated.

To overcome this problem, the Extensible Authentication Protocol [13] was designed as a new PPP authentication framework to enable a PPP client to authenticate with a back-end EAP server without the NAS needing to know the details of the authentication method. The NAS simply passes through EAP packets between the PPP client and the EAP server, until it gets a success or a failure indication from the EAP server. In principle the EAP server functionality can be co-located in the NAS but typically the EAP server is a separate entity with which the NAS exchanges EAP packets over Radius or Diameter. In brief, EAP specifies a framing for multi-roundtrip authentication protocols.

During the EAP authentication, the EAP server sends EAP Requests to the client. The EAP Type field in the EAP packet indicates what it being requested. The client sends an EAP Response of the requested type in reply to each EAP Request. The first EAP Request is usually of the type Identity. It can be issued by the NAS even if a separate EAP server is used. The client's response to this request contains the client's identity. User identities of the Network Access Identifier (NAI) format [14], which contains information to find the correct AAA server, are used in roaming environment.

The Identity Request is followed by one or more EAP Requests for authentication information. Eventually, the EAP server ends the authentication with an EAP Success or an EAP Failure packet.

The EAP Type field in the EAP packet distinguishes different authentication protocols. EAP encapsulations have been specified or are being specified for a number of protocols, including MD5 Challenge (CHAP), one-time passwords, Transport Layer Security (TLS), Secure Remote Password (SPR), Generic Security Services (GSS) API, GSM authentication and UMTS authentication.

In PPP, the client never re-transmits an EAP packet on a timer. It is the responsibility of the Network Access Server to retransmit the EAP Request if it receives no response from the client.

NAAP re-uses the PPP EAP framework and the EAP AAA message formats. This makes it possible to use existing AAA servers, AAA protocols and EAP types with minimal modifications.

Some EAP types are capable of distributing a session key as part of the authentication procedure. TLS, SRP, GSM and UMTS authentication protocols are examples of such EAP types. Because EAP was originally designed for physically secure channels, NAAP is not just a simple EAP over UDP protocol. NAAP includes additional functionality to protect the NAAP messages against various attacks that may take place on a wireless shared-medium network. For this functionality, NAAP makes use of session keys distributed by the EAP type. Therefore, NAAP requires that the EAP type used must distribute a session key.

3. OVERVIEW ON NAAP AUTHENTICATION

A NAAP Client uses NAAP agent discovery to learn of local NAAP Agents. Agent discovery resembles Mobile IP agent discovery. It consists of two NAAP messages, the NAAP Solicitation, which a NAAP Client may multicast or broadcast, and the NAAP Advertisement, sent by a NAAP Agent to a NAAP Client in response to client's NAAP solicitation. Analogously to Mobile IP, the NAAP Advertisement includes the IP address of the NAAP Agent and the minimum and maximum registration lifetimes supported.

If the client discovers that the current network employs a NAAP Agent, the client needs to register with the agent in order to get network access. A NAAP registration is the exchange of NAAP Request and NAAP Reply messages. The NAAP Reply message contains a Code field, which tells the outcome of the registration. For example, the Code value SUCCESSFUL says that the terminal was registered successfully. During the EAP authentication, the NAAP Client and the NAAP Agent need to exchange several NAAP Requests and NAAP Replies before the client is successfully authenticated. The NAAP Replies that require the NAAP Client to send more NAAP Requests use the Code value AUTHENTICATION_INCOMPLETE.

Figure 19 illustrates the NAAP authentication when the terminal enters an access network. First the terminal discovers the NAAP Agent by sending a NAAP Solicitation message and receiving a NAAP Advertisement message from the agent.

The NAAP Client starts the EAP authentication after receiving the NAAP Advertisement. The first NAAP Request the terminal sends includes an EAP extension, which contains the EAP Response packet of the type

Identity. As usual in EAP, the EAP-Response/Identity packet contains the NAAP Client's AAA identity, the Network Access Identifier (NAI). The EAP-Request/Identity packet is not used in NAAP, but the EAP authentication always begins with the client's NAAP Request that includes EAP-Response/Identity.

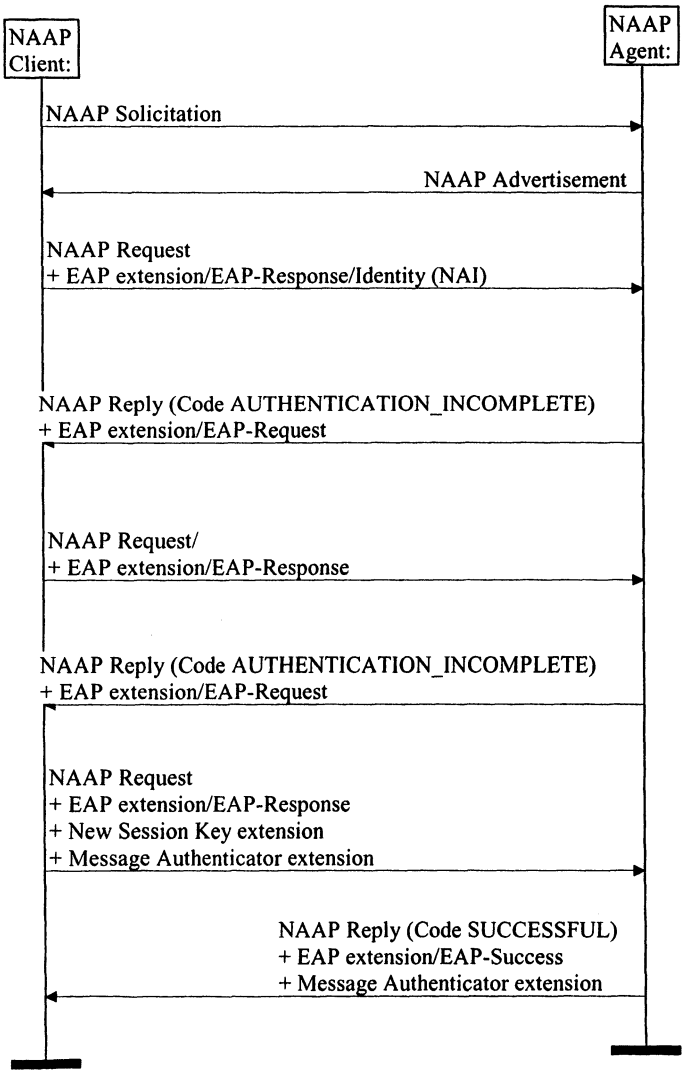

Figure 19. Successful NAAP authentication

Following the NAAP Request that contains EAP Response/Identity, the NAAP Agent sends the terminal several EAP Requests (in NAAP Reply

packets) and the terminal replies with NAAP Requests that contain the corresponding EAP Responses. These EAP packets contain the actual authentication information.

The EAP authentication results in that a new session key is generated and becomes shared by the NAAP Client and the NAAP Agent. The NAAP Client derives the key using the information in the EAP Request packets, and the NAAP Agent receives the key from the AAA server.

The key is associated with a Security Parameter Index (SPI) number that identifies the key. It is useful in distinguishing an old key from a new key that has been derived during a re-authentication. The NAAP Client chooses the SPI value and transmits it to the NAAP Agent in the last NAAP Request shown in *Figure 19*. The last NAAP Request contains the last EAP Response. This NAAP Request message must include the Authenticator extension (Section 5) calculated with the new session key. The NAAP Request also includes the New Session Key extension, which specifies how the new session key will be used. For example it contains the SPI value picked up by the client. In a successful case, the AAA response the NAAP Agent gets from the Authentication Server concerning this message includes the EAP Success packet and the new session key. The NAAP Agent verifies the message authentication code (MAC) in the Authenticator extension and replies with the NAAP Reply containing the EAP Success packet from the AAA response.

If the authentication fails or if an error occurs on the EAP server, the NAAP Agent sends the NAAP Client an EAP Failure packet in a NAAP Reply with the Code FAILURE. The AAA message from the AAA server may contain an attribute that contains an error message to the user. The NAAP Agent may include the error message as a Reply-Message extension in the NAAP Reply that it sends to the NAAP Client. *Figure 20* shows an example of an unsuccessful authentication.

A design principle in NAAP is "smart terminal, dumb network". By keeping the NAAP Agent workload per each client as small as possible the agent is expected to be able to serve more clients. Hence, the NAAP Client is responsible for retransmissions and soft state refreshing. The retransmissions in NAAP are different from EAP over PPP, where the Network Access Server is responsible for retransmissions.

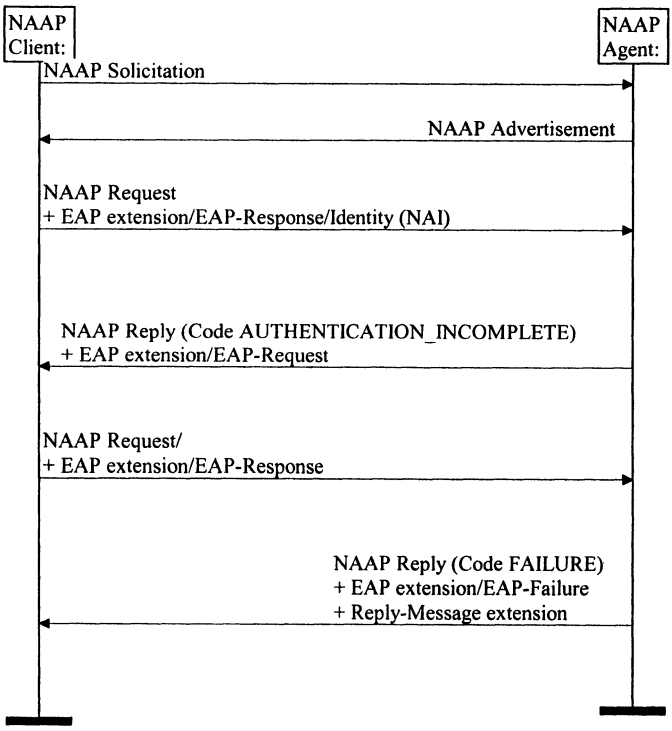

Figure 20. Unsuccessful NAAP authentication

4. LOCAL REGISTRATIONS AND DEREGISTRATION

After the full AAA registration, the NAAP Client and the NAAP Agent may use local NAAP messaging to extend the registration lifetime, to disconnect and to enable the agent to detect silently disconnected clients.

The NAAP Client maintains two timers: a preferably short-term registration lifetime timer, which makes sure that the terminal periodically refreshes its NAAP registration, and a typically long-term EAP authentication timer, which makes sure that the full EAP authentication and key exchange is executed periodically. As explained in Section 6, local NAAP messages cannot always be used. In this case, the registration lifetime timer is also a long-term timer and the client does not need to frequently refresh its NAAP registration.

The local NAAP registration, illustrated in *Figure 21*, occurs locally between the terminal and the agent to refresh the NAAP registration. When local registrations are used, a short registration lifetime, for example 60

seconds, is chosen. If the terminal does not re-register during the registration lifetime, the agent can deduce that the terminal has left the access network. Knowing when the terminal leaves is important if the network accounts for the connection time.

The NAAP Client may explicitly close the NAAP session by sending a local NAAP Request with the requested lifetime field set to zero. The agent responds to these deregistration requests with a NAAP Reply. The formats of the de-registration NAAP Request and the corresponding NAAP Reply are similar to the local re-registration messages.

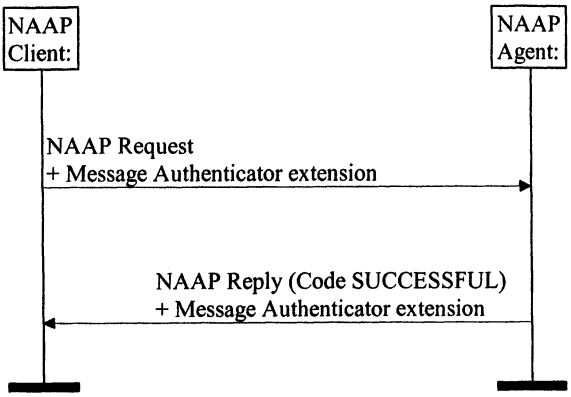

Figure 21. A local NAAP registration

5. PROTECTION AGAINST MESSAGE TAMPERING AND REPLAY

The EAP type used in NAAP must be strong enough so that the EAP packets can be transmitted in the clear during EAP authentication. The first NAAP messages do not contain any cryptographic protection. Once a session key has been distributed, the Authenticator extension can be included in a NAAP Request or a NAAP Reply message to protect the message against tampering. The Authenticator extension contains a Message Authentication Code (MAC) that is calculated over the NAAP message using the key that the NAAP Client and the NAAP Agent share as a result of EAP authentication. The MAC is calculated as a HMAC keyed hash [15] using the MD5 hash function [16]. Besides the MAC, the Authenticator extension contains the Security Parameter Index that identifies the session key.

The Authenticator extension must be included in two cases:

1) In an EAP authentication and re-authentication, the NAAP Request that contains the last EAP Response must include the New Session Key extension and the Authenticator extension calculated using the new key. The corresponding NAAP Reply must also be protected with an Authenticator extension calculated using the new key. The purpose of the message authentication in this case is to protect against a man-in-the-middle attacker from changing the NAAP fields outside the EAP packet. For example, the NAAP Agent ascertains that the IP address of the NAAP Client is correct.

2) Local NAAP Requests and Replies that do not contain an EAP extension must include the Authenticator extension. This includes the local registrations that are performed to extend the lifetime of the NAAP registration, and deregistrations that are performed to disconnect. This prevents a malicious node from hijacking the client's session after the client has left the network. It is also impossible to mount a denial of service attack by simply sending a deregistration request on behalf of another terminal.

Some error messages in NAAP do not include the Authenticator extension. If the authentication fails, no key is distributed and hence the NAAP packet that contains EAP Failure cannot be protected. Therefore it is easy to mount a denial of service attack by injecting a false EAP Failure packet to the client. The NAAP Client protects against these attacks by accepting unauthenticated error messages only after the retransmission timer has been expired. If a correct message is received before the timer expires, the false error message is ignored.

Another way to handle these cases would be to only send error messages if they can be authenticated and let timer expiration take care of other error cases. In NAAP, unauthenticated error messages are used, however, because they can contain a potentially useful error message. For example, the user may be told that the visited network does not have a roaming agreement with the user's home service provider, or that his connections are barred because of unpaid bills.

In order to prevent an attacker from replaying old NAAP registration messages, NAAP protects against replay attacks with a timestamp in the NAAP Request and NAAP Reply. The replay protection is required for the local NAAP registrations that do not involve the AAA network, because the authentication of these messages is only based on the message authentication code in the Authenticator extension.

Usually in protocols, protection against replay attacks is based on either timestamps or the exchange of random values. In the latter alternative, one of the parties sends a fresh random number to the other party, which replies with a message that include a value calculated using the random number and a shared key. In NAAP, replay protection is based on timestamps and it is similar to Mobile IP timestamp replay protection. The basic principle is that

the NAAP Client inserts the current time of day in the NAAP Request message, and the NAAP Agent receiving the message checks that this timestamp is sufficiently close to its own time of day. Because the timestamp is included in the calculation of the Authenticator extension, an attacker cannot make an old message current just by modifying the timestamp. Obviously the two nodes must have adequately synchronized time-of-day clocks. The timestamp difference tolerance is a configurable parameter on the agent. The NAAP Request and NAAP Reply messages can be used to synchronize the clock on the mobile terminal. Time synchronization NAAP messages are always protected against tampering with the Authenticator extension, so that an attacker cannot fool the client to adjust its clock to wrong time. This could enable the attacker to obtain NAAP Requests with timestamp values that are in the future, and then use these requests to hijack the client's session when the timestamps become current.

Upon receipt of a NAAP Request with the Authenticator extension, the NAAP Agent checks the timestamp for validity. In order to be valid, the timestamp must be close enough to the NAAP Agent's time of day clock and the timestamp must be greater than all previously accepted timestamps for the requesting NAAP Client. If the timestamp is not valid, the NAAP Agent includes a timestamp from its own time of day. The NAAP Agent rejects the registration by returning a Code that indicates incorrect timestamp in the NAAP Reply.

The NAAP Agent does not check the timestamp for validity if the NAAP Request does not include the Authenticator extension. In these cases, the field is only used for matching NAAP Replies to Requests.

If a NAAP Client retransmits a NAAP Request, it uses a new timestamp value containing the current time of day.

6. NAAP AND VPN SOFTWARE

Users of wireless LANs often use a Virtual Private Network solution to provide for user data confidentiality. When a VPN connection is active, all IP traffic is typically tunneled to a VPN gateway. The VPN client software typically includes a configurable security policy with which the user can specify the processing required for different types of traffic. Ideally, the policy file should allow local exchange of DHCP and NAAP packets so that these protocols are not tunneled to the VPN gateway.

Sometimes the user is not allowed or willing to modify the VPN policy. We discovered during the system testing that if the VPN software tunnels all traffic to a remote VPN gateway, NAAP packets will not reach the NAAP Agent after a VPN connection has been activated. In these cases, DHCP

leases or NAAP registrations cannot be renewed, which means that communications will fail when the NAAP registration or the DHCP lease expires.

If local NAAP messages cannot be used, NAAP terminal presence detection based on NAAP local registrations is not available. To support this scenario, NAAP protocol was enhanced so that the agent may advertises a long maximum registration lifetime in the NAAP Advertisements. The NAAP Client chooses a registration lifetime between the advertised minimum and maximum, and supplies it in the lifetime field of the NAAP Request packet. If the client chooses to use a short registration lifetime, the agent can rely on the NAAP registrations for terminal presence detection. However, if the terminal chooses to use a long registration lifetime, the agent may try to detect inactive terminals for example with a an ARP request or an ICMP echo request (ping) after the terminal has been idle for some time. If the terminal does not reply to a couple of successive ARP request or ICMP echo request, the agent may deduce that the terminal has left the access network. Relying on ARP or echo replies makes it possible to hijack the session of a terminal that has left the network, if the hijacker assumes the terminal's IP and MAC addresses and continues using the network on the original user's account. Therefore NAAP Client should use local registration whenever possible.

7. CONCLUSIONS

The NAAP protocol described in this paper has been successfully implemented in the Nokia Operator Wireless LAN product. The author is participating in the standardization of an IP layer user-to-network authentication protocol (User Registration Protocol, URP), which is currently beginning in the Internet Engineering Tack Force (IETF).

NAAP is a rather straightforward registration protocol and therefore easy to implement even on current operating systems. However, during the system testing, some problems in the co-existence of NAAP and Virtual Private Networking software were discovered, and the NAAP protocol was enhanced to work around these problems.

Currently, the NAAP Agent implementation enforces the access control by filtering out the packets of unauthenticated clients based on IP and link-layer addresses. The keying material distributed in NAAP could easily be used for packet security between the client machine and the router that contains the NAAP Agent. For example, there could be an IP security (IPsec) tunnel between the client machine and the router. Packet

authentication and encryption would provide for better access control and user data confidentiality.

ACKNOWLEDGEMENTS

Thanks to Antti Kuikka, Jukka Latva, Lassi Lehtinen, Jyri Rinnemaa, N. Asokan and all others who have contributed in the development of the NAAP protocol.

REFERENCES

[1] Nokia Operator wireless LAN WWW page, http://www.nokia.com/networks/wireless_lan/index.html, August 2001

[2] W. Simpson, Editor, " The Point-to-Point Protocol (PPP)", IETF Request For Comments 1661, July 1994

[3] C. Rigney, S. Willens, A. Rubens, W. Simpson, "Remote Authentication Dial In User Service (RADIUS)", IETF Request For Comments 2865, June 2000.

[4] C. Rigney, "RADIUS Accounting", IETF Request For Comments 2866, June 2000.

[5] C. Rigney, W. Willats, P. Calhoun, "RADIUS Extensions", IETF Request For Comments 2869, June 2000.

[6] The Diameter protocol is being specified in the IETF and it is currently work in progress. Current versions of the Diameter documents are available at http://www.ietf.org/html.charters/aaa-charter.html (October 2001)

[7] C. Perkins, "IP Mobility Support", Internet Engineering Task Force, Request For Comments 2002, 1996.

[8] C. Perkins, "IP Encapsulation within IP", Internet Engineering Task Force, Request For Comments 2003, 1996

[9] C. Perkins, "Minimal Encapsulation within IP", Internet Engineering Task Force, Request For Comments 2004, 1996

[10] J. Solomon, "Applicability Statement for IP Mobility Support", Internet Engineering Task Force, Request For Comments 2005, 1996

[11] D. Cong, M. Hamlen, C. Perkins, "The Definitions of Managed Objects for IP Mobility Support using SMIv2", Internet Engineering Task Force, Request For Comments 2006, 1996

[12] G. Montenegro, "Reverse Tunneling for Mobile IP", Internet Engineering Task Force, Request For Comments 2344, 1998

[13] L. Blunk, J. Vollbrecht, "PPP Extensible Authentication Protocol (EAP)", IETF Request for Comments: 2284, March 1998

[14] B. Aboba and M. Beadles, "The Network Access Identifier", Internet Engineering Task Force, Request For Comments 2486, January 1999.

[15] H. Krawczyk, M. Bellare, R. Canetti, "HMAC: Keyed-Hashing for Message Authentication", Internet Engineering Task Force, Request For Comments 2104, February 1997

[16] R. Rivest, "The MD5 Message-Digest Algorithm", Internet Engineering Task Force, Request For Comments 1321, April