available at www.sciencedirect.com

ScienceDirect

journal homepage: www.elsevier.com/locate/cose

Computers & Security

OSNP: Secure wireless authentication protocol using one-time key

Y.L. Huang[a,*], P.H. Lu[a], J.D. Tygar[b], A.D. Joseph[b]

[a]Dept. of Electrical & Control Eng., National Chiao-Tung Univ., Hsinchu, Taiwan
[b]Dept. of Electrical Eng. & Computer Sciences, Univ. of California, Berkeley, USA

ARTICLE INFO

Article history:
Received 8 March 2009
Received in revised form
26 April 2009
Accepted 15 May 2009

Keywords:
802.11 WLAN security
Handover security
Handover authentication
Mutual authentication
Inter-domain authentication
EAP-based protocol
SWOON experiments

ABSTRACT

Handover security and efficiency have become more and more important in modern wireless network designs. In this paper, we propose a new protocol using the one-time key for user authentication. The proposed protocol can support both intra-domain and inter-domain authentications efficiently. Our protocol requires five messages for intra-domain initial authentication; three for subsequent authentication; and five for handover authentication. No authentication server is needed during handover, and our design reduces the computing load on the authentication server. We show an integration and implementation of EAP from 802.1X and our protocol, giving an easy way to apply our protocol on existing 802.11 wireless networks. The proposed protocol is realized and verified on the SWOON secure wireless testbed.

1. Introduction

To enhance the security in wireless networks, IEEE 802.11i (Ieee, 2003) defines a new security model for 802.11 a/b/g networks, specified new standards for authentication, encryption and message integrity, and implemented 802.1X (Ieee, 2001) for user authentication and key distribution. 802.1X is a port-based network access control mechanism that provides an extensible authentication protocol (EAP) (Aboba et al., 2004) and can be used in conjunction with other popular authentication protocols, such as TLS, PEAP, CHAP, etc.

There are many EAP methods supporting 802.11i authentication, including EAP-TLS (Ppp Eap, 1999), EAP-FAST (The Flexible Authentication, 2007), and LEAP (Macnally, 2001). EAP-TLS and EAP-FAST are used with public-key cryptography for authentication. Compared to symmetric-key systems, public-key systems and certificates ensure stronger security, but require more computational power. LEAP, a symmetric-key authentication protocol, requires less computational power and thus takes less response time when performing user authentication. However, LEAP is vulnerable to several attacks (Baek et al., 2004) such as weak encryption keys. To balance the efficiency and security, an efficient authentication is required for wireless networks, especially for roaming users.

In 2007, Zrelli et al. (Zrelli and Shinoda, 2007) presented an integration of the Kerberos protocol with EAP framework, called EAP-Kerberos. Kerberos (Neuman et al., 2005) is well known for its symmetric-key cryptography, strong per-person key and inter-domain authentication. However, it is still

* *Corresponding author*. Tel.: +886 3 5131476; fax: +886 3 5715998.
E-mail addresses: ylhuang@cn.nctu.edu.tw (Y.L. Huang), watil.eceforever@gmail.com (P.H. Lu), doug.tygar@gmail.com (J.D. Tygar), adj@eecs.berkeley.edu (A.D. Joseph).
0167-4048/$ – see front matter © 2009 Elsevier Ltd. All rights reserved.
doi:10.1016/j.cose.2009.05.009

inefficient in the WLAN environment because users need to use proxies to get tickets from the Kerberos Key Distributed Center (KDC). In other words, in Kerberos, KDC is involved in the handover of a roaming user.

In this paper, we propose a handover authentication protocol for WiFi (802.11) networks. Our protocol does not require a public key infrastructure and can be integrated with the 802.1X (Ieee, 2001) Extensible Authentication Protocol (EAP) (Aboba et al., 2004). The novel contributions of this paper are:

- We propose an efficient authentication protocol supporting handover authentication without a trusted of third-party.
- We integrate the proposed protocol with the EAP framework so that it can be applied to wireless network authentication with only minor efforts.
- We implement the proposed protocol and verify it using the SWOON testbed, a secure wireless network emulation testbed. We also compare it with other EAP methods in terms of communication, computation and storage costs.

The rest of this paper is organized as follows. Related research is detailed in Section 2. We present our authentication protocol and the integration with EAP in Sections 3 and 4, respectively. Section 5 compares our system to others, and Section 6 concludes the paper. In the Appendix, we use BAN logic (Burrows et al., 1989) and a possible enhancement (Shieh et al., 1999) to formally prove that our protocol can reach the goals of mutual authentication.

2. Related work

2.1. *Network authentication protocols*

This section summarizes the characteristics and drawbacks of some related authentication protocols.

- Kerberos

  Kerberos (Neuman et al., 2005) was developed as a solution to network security problems, such as replaying, eavesdropping and sniffing packets. In Kerberos V5, six messages are required for initial intra-domain authentication. The number of message required for inter-domain authentication depends on the number of KDCs between the visited and home domains.

- One-Time Password/Kerberos

  Since the traditional password authentication is vulnerable to dictionary and playback attacks, in 2005, Cheng et al. (Xiao-rong et al., 2005) presented a new authentication method that integrates the Kerberos protocol and a one-time password (OTP) system. The main idea of OTP authentication is to add random factors during the initial login process and make the password used vary from time to time. Similar to the Kerberos protocol, the OTP/Kerberos protocol requires three steps to authenticate a user: authentication by the KDC, request of tickets from the Ticket-Granting Server (TGS) and access to the server (S). On the client, the OTP is generated by hashing the user's secret passphrase and the seed from the KDC. By encrypting and decrypting messages with the OTP, the user and server mutually authenticate each other. However, to generate an OTP for authentication requires seven messages in the first step mentioned above. In other words, OTP/Kerberos increases the communication cost for authenticating a user, resulting in longer user authentication times, which is not practical for roaming users in wireless networks.

- Secure Authentication Protocols

  In the last decade, many secure authentication protocols (Shieh et al., 1999; Chien, 2003; Liang and Wang, 2004; Hwang, 2005) were proposed to solve the issues of Kerberos. Among these protocols, Secure Network Protocol (SNP) (Shieh et al., 1999) is one of the few protocols that has real deployment for years. SNP is a symmetric-key based protocol providing an efficient way for both intra- and inter-domain authentication. Compared to Kerberos, fewer messages are required in SNP to authenticate client identity. For intra-domain authentication, SNP takes four messages to authenticate client identity and one more optional message for mutually authenticating the server. For inter-domain authentication, it takes seven messages for initial authentication, regardless of the number of hops between the visited and home domains. Only two messages are required for subsequent authentication when requesting the same service. To simplify the design, SNP replaces timestamps with nonces, reducing the need for time servers. For faster authentication, a master key is shared by the authentication server (AS) and the service servers (S). The unchanged master keys can make the system vulnerable to various attacks.

2.2. *EAP-based authentication protocols*

EAP is an authentication framework used in various networks, such as wireless LANs and Point-to-Point connections (PPP). EAP provides some common functions and a negotiation of the desired authentication methods, such as EAP-MD5, EAP-OTP, EAP-TLS, etc. In wireless LANs, EAP authentication methods are normally supported with Remote Authentication Dial-In User Service (RADIUS) (Remote Authentication Dial, 2000). RADIUS is also a client/server protocol that enables remote access servers to communicate with a centralized authentication server to authenticate dial-in users. It also authorizes their access to the requested services. The RADIUS server supporting various EAP methods then becomes the major authority of wireless networks. This section summarizes EAP authentication methods supporting strong authentication for roaming users in wireless LANs.

- EAP-TLS

  EAP-TLS (Ppp Eap, 1999) is a popular EAP method for securing wireless LANs with RADIUS. The mobile node and RADIUS server must have certificates to mutually authenticate each other. EAP-TLS is resilient to man-in-the-middle attacks. However, it requires a trusted-third party (Certificate Authority) to support authentication between the

mobile node and RADIUS server. Also, it requires extra management for administrating and distributing certificates, supported by cooperative network management systems (NMS) or Operation, Administration, Maintenance and Provisioning (OAM&P).

- EAP-Kerberos

  In 2007, S. Zrelli and Y. Shinoda (Zrelli and Shinoda, 2007) showed how to integrate the Kerberos protocol as an authentication method in EAP-based authentication frameworks. They define the architectural elements and specify the encapsulation of Kerberos messages in EAP packets. Such a design allows a mobile node to be authenticated using the Kerberos systems. When a mobile node, for example, issues an initial authentication request, the EAP-Kerberos client encapsulates the Kerberos messages into EAP packets and sends them to the access node. The access node then delivers these EAP packets to the RADIUS server via the AAA (Authentication Authorization and Accounting) protocol. The server either validates these messages or forwards them to the Kerberos KDC, as shown in Fig. 1.

- Kerberized Handover Keying (KHK)

  In 2007, Ohba et al. (2007) proposed a Kerberized media-independent handover key management architecture for the existing link-layer technologies, including 802.11 and 802.16. The architecture uses Kerberos for securing key distribution between a mobile node, an access point (authenticator) and a server. In KHK, two handover modes are presented: proactive and reactive. In proactive mode, a mobile node uses a pre-obtained credential to authenticate with the access point. In reactive mode, the access point acts as a Kerberos client on behalf of the mobile node. In this architecture, Kerberos can be bootstrapped from initial authentication using an EAP method. This makes KHK work across multiple AAA domains. However, similar to Kerberos, the KDC is involved in handover authentication in KHK. Thus, the handover performance for reactive mode depends on the location of the KDC. The larger the distance between a KDC and a mobile node, the longer the time required for a handover authentication. Also, such an architecture incurs extra costs for setting up time servers for synchronizing machine times in the network, as mentioned in the previous section.

3. Proposed protocol: OSNP

In this paper, we propose the integration of SNP, OTP and EAP for authenticating IEEE 802.11 mobile nodes, giving us support for fast roaming—One-time key Secure Network Protocol (OSNP).

3.1. *Preliminaries*

Table 1 shows the abbreviations and symbols used in our protocol.

Similar to other password-based authentication methods, our authentication servers or KDCs share secrets with users and servers in their own domains. For example, the user chooses his own strong password and shares it with his KDC; the server chooses its own strong password and shares it with its KDC. These shared secrets are assumed to be stored in a secure storage system.

3.2. *Intra-domain authentication*

In our protocol, we give three methods for three types of intra-domain authentication: initial, subsequent and handover authentication. In initial authentication, five messages are required for mutually authenticating the user and server. To subsequently authenticate with the same server, only three messages are used (we renew session keys without querying the KDC.) Handover authentication requires five messages to renegotiate a new session key with another server of the domain. Although five messages are required, the KDC is not involved, reducing its load.

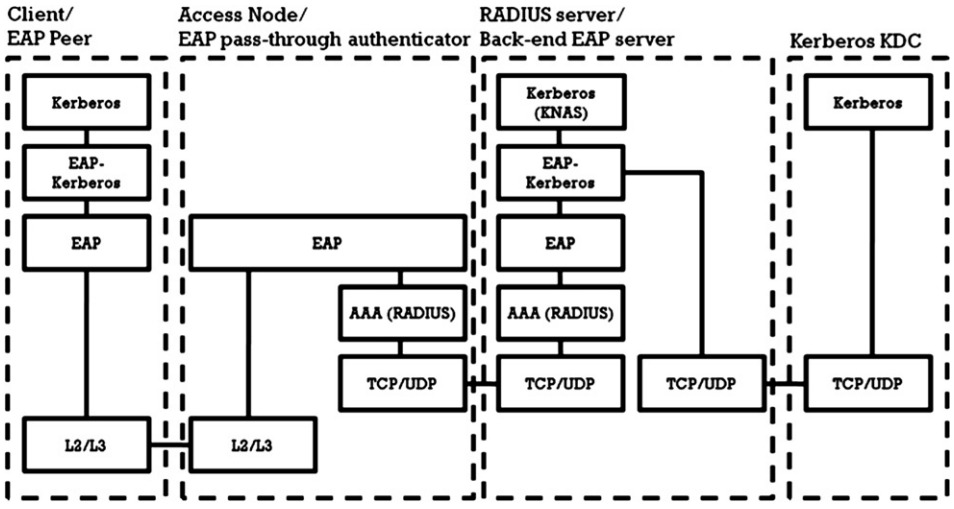

Fig. 1 – Component protocol stacks of the EAP-Kerberos authentication framework.

Table 1 – Message abbreviations.

Abbreviation	Description
TKT_x	Ticket issued by X
CH_x	Challenge issued by X
$RESP_x$	Response to CH_x
A_x	Authenticator issued by X
$authRQ_x$	Authentication request sent by entity X
$authAK_x$	Authentication response to $authRQ_x$
$sauthRQ_x$	Subsequent authentication request issued by X
$sauthAK_x$	Authentication response to $sauthRQ_x$
$hauthRQ_x$	Handover authentication request issued by X
$hauthAK_x$	Authentication response to $hauthRQ_x$
$hquthVF_x$	The verifier sent by the previous server S_{old}
$iauthRQ_x$	Inter-domain authentication request issued by X
$iauthAK_x$	Authentication response to $iauthRQ_x$
$iauthFW_x$	Inter-domain authentication forwarding to X
U_a	User principal in domain "a"
S_a	Server principal in domain "a"
KDC_a	Key distribution center in domain "a"
PW_x	Password of entity X
N_x	Nonce generated by X
K_{ss}	Session key for securing the communication
K_g	Group key for all servers under a KDC
K_{TU}	Temporal user key for subsequent authentication
OTK_x	One-time key of entity X
SID	Session identification
VT_x	Validation time (Expiration) of entity X
T_{Ux}	Temporary user identity of entity X
rt	The remaining time of the validate ticket
ct	The current time of the local host

Since authentication occurs in a common domain, the notations U_a, S_a, and KDC_a are simplified to U, S, and KDC, respectively.

3.2.1. *Initial authentication*

To initially access a server, the wireless client U sends the authentication request message (M1) to the server. The message is then forwarded to the KDC with server credentials (M2). The KDC authenticates the identities of user and server; generates a session key and sends the message (M3) back to the server. The server then forwards the message with encrypted session key (M4) to the user. The final acknowledge (M5) is sent back to the server for mutual authentication.

M1. U → S: $authRQ_U$
M2. S → KDC: $authRQ_S||authRQ_U$
M3. KDC → S: $SID||authAK_S||authAK_U$
M4. S → U: $authAK_U||CH_S||TKT_S$
M5. U → S: $RESP_S||A_U$

M1. When the authentication process starts, the user generates an authentication request, containing a nonce N_U, the user's identity and an encrypted message using user's one-time key OTK_U: $authRQ_U = U||N_U||\{U, N_U\}_{OTK_U}$. The one-time key is a hashed value of user's identity, password and nonce: $OTK_U = Hash(U, N_U, PW_U)$. The request is sent to the server for authentication.

M2. After receiving the user request, S generates its request $authRQ_S = S||N_S||\{S, N_S\}_{OTK_S}$. It then concatenates the two requests and sends them to the KDC.

M3. To identify each session, the KDC generates a unique identity $SID = U|S|N_U$ for each session after receiving the authentication requests. It also calculates the one-time keys OTK_U and OTK_S to verify the requests. After authenticating both identities, the KDC randomly generates a session key K_{SS}. The session key, server's nonce, and identity are encrypted with OTK_S to acknowledge the server's authentication request.

$$authAK_S = \{N_S, U, K_{SS}\}_{OTK_S}$$

A temporary user key K_{TU} is generated and encrypted with OTK_U in the acknowledgement.

$$authAK_U = \{N_U, S, K_{SS}, K_{TU}\}_{OTK_U}$$

The temporary user key can be used for subsequent authentication.

M4. Upon receiving the response from KDC, S decrypts $authAK_S$ in M3 with OTK_S and gets the session key. The server generates a new challenge for authenticating the user. The new challenge is made by encrypting a new nonce N'_S and the server's identity with the session key K_{SS}, represented as $CH_S = \{S, N'_S\}_{K_{SS}}$. In addition, S can also optionally generate a ticket (TKT_S) for subsequently authenticating the same user.

$$TKT_S = SID \parallel \{U, VT_S, K_{SS}\}_{OTK_S}$$

VT_S is a validation time for TKT_S. Since the validation of a ticket is determined by its issuer, no time server is required.

M5. The user receives the session key after decrypting $authAK_U$. It then generates a response $RESP_S = \{U, N'_S\}_{K_{SS}}$ to CH_S. The response is generated by replacing the server's identity with the user's. Then, the mutual authentication of the user and server can be guaranteed by encrypting and decrypting these messages with the shared session key. In addition, a temporary authenticator $A_U = \{S, VT_U, K_{SS}\}_{K_{TU}}$ is also appended to the response message. The authenticator can be used to authenticate the user in subsequent authentication rounds without querying the KDC. As above, no time server is needed.

3.2.2. *Subsequent authentication*

Subsequent authentication rounds occur when a user requests the same services within the specified time. For intra-domain subsequent authentication, the user must send the ticket and his temporary credential to the server. Below is the flow for intra-domain subsequent authentication.

M1. U → S: $sauthRQ_U$
M2. S → U: $sauthAK_U||CH_S||A_U$
M3. U → S: $RESP_S$

M1. To initiate a subsequent authentication, the user generates a subsequent authentication request, consisting of a nonce and a ticket, and sends it to the server. The subsequent authentication request can be represented as $sauthRQ_U = N_U||TKT_S$.

M2. After receiving the $sauthRQ_U$, the server retrieves the user identity from the SID contained in TKT_S. Then, the server decrypts the ticket and checks its validation time VT_S. If the ticket is not expired, the server generates a nonce and a new session key. The user nonce and a new session key are then encrypted with the previous session key to acknowledge the request from user.

$sauthAK_U = \{N_U, K'_{SS}\}_{K_{SS}}$

Then, the server nonce and identity are encrypted with the new session key. This is a new challenge for mutually authenticating the user.
$CH_S = \{S, N_S\}_{K'_{SS}}$

A concatenation of $sauthAK_U$, CH_S with the temporary authenticator A_U received in the initial authentication is then sent back to the user.

M3. The user decrypts the temporary authenticator A_U to get VT_U and K_{SS}. It checks the validation time of the temporary authenticator, if the authenticator is not expired, the user decrypts $sauthAK_S$, and obtains the new session key. The nonce N_S and the user identity are then encrypted using the new session key to respond the CH_S. The subsequent response is represented as $RESP_S = \{U, N_S\}_{K'_{SS}}$.

3.2.3. Handover authentication

Handover authentication occurs when a user requests a server belonging to the same domain as the previous server. In most network authentication protocols, an initial authentication is required when contacting another server. This increases the load on the KDC. In such a case, since the user is already authenticated by the KDC and recognized by the previous server, re authentication can be performed by the previous server to reduce the load on the KDC. In this paper, we propose a 5-step handover authentication protocol for intra-domain authentication.

M1. U → S: $hauthRQ_U$
M2. S → S_{old}: $CH_S \| hauthRQ_U$
M3. S_{old} → S: $SID \| hauthVF_{S_{old}}$
M4. S → U: $hauthAK_U \| CH_S \| TKT_S \| A_{U_{old}}$
M5. U → S: $RESP_S \| A_U$

M1. Similar to subsequent authentication, a user sends a $hauthRQ_U$ to initiate a handover authentication. The $hauthRQ_U$ is the same as $sauthRQ_U$, containing a user identity, nonce and ticket to the previous server: $hauthRQ_U = U \| N_U \| TKT_{S_{old}}$.

M2. S generates a $CH_S = \{S, N_S\}_{K_g}$ and forwards it together with the $hauthRQ_U$ to its previous server S_{old} in $TKT_{S_{old}}$.

M3. After validating the ticket, S_{old} retrieves the user identity, validation time $VT_{S_{old}}$ and the previous session key $K_{SS_{old}}$ from the $TKT_{S_{old}}$. The server S_{old} then calculates the remaining validation time for the ticket: $rt = VT_{S_{old}} - ct$. The remaining validation time, user identity and the previous session key are encrypted together with the response to CH_S and the temporary authenticator $A_{U_{old}}$ using the group key K_g. $hauthVF_{S_{old}} = \{U, K_{SS_{old}}, rt, A_{U_{old}}\}_{K_g}$ is sent to S securely.

M4. Upon receiving M3, S decrypts the message with the group key and gets the previous session key, the temporary authenticator and the remaining validation time of the previous ticket. The temporary authenticator will be forwarded to the user for proving the user's identity. When generating the new ticket TKT_S for the user, S calculates its validation time according to the remaining validation time.

$VT_S = ct + rt$

An acknowledgement $hauthAK_U = \{N_U, K_{SS}\}_{K_{SS_{old}}}$ is generated responding to $hauthRQ_U$ in M1. Also, a challenge $CH_S = \{S, N'_S\}_{K_{SS}}$ is sent to the user for mutual authentication.

M5. As above, the user decrypts messages to get the new session key and his temporary authenticator A_U. The new session key is used to generate the response $RESP_S = \{U, N'_S\}_{K_{SS}}$ to the CH_S and the new temporary authenticator $A_U = \{S, VT_U, K_{SS}\}_{K_{TU}}$.

3.3. Inter-domain authentication

The proposed inter-domain authentication takes advantages of the SNP design. All KDCs in the hierarchy share keys. This reduces the time required for querying and searching to locate the home KDC of the visiting user. A user TU_x roaming from domain X, for example, wants to access a server S_Y in a foreign domain Y. His authentication request will be sent to S_Y and then to the foreign KDC_Y. Since KDC_Y cannot authenticate the user, the authentication request will be forwarded back to the previously visited KDC_x after KDC_Y locates the KDC_x. In our proposal, a root KDC_R identifies a previously visited KDC for a foreign KDC. Once TU_x is authenticated by KDC_x, the user TU_x will receive a temporary identity TU_Y for its subsequent services in the domain Y. Fig. 2 illustrates the initial authentication flow for an inter-domain authentication.

3.3.1. Hierarchical KDC

In the previous section, we presented an authentication protocol for authenticating users who registered in the same security domain. However, for a very large network, it is impractical for all the users to be registered in a single domain. Instead, users and servers should register with their own KDCs, which form a hierarchical structure. In such a structure, each node in the hierarchy represents a domain, where parent domains manage all their children domains. Each domain has one KDC to manage the authentication of its users and servers.

In the proposed inter-domain authentication protocol, every KDC must share a different secret key with all its ancestors to perform inter-domain authentication efficiently. Consequently, the root KDC needs a large database to store the shared keys for all descendant KDCs. Fortunately, the size of a key is small, and the root KDC is able to store all the keys.

3.3.2. Protocol description

Similar to the intra-domain initial authentication, our approach starts with a request from the user from a foreign domain.

M1. TU_x → S_Y: $authRQ_{TU_X}$
M2. S_Y → KDC_Y: $authRQ_{S_Y} \| authRQ_{TU_X}$
M3. KDC_Y → KDC_R: $iauthRQ_{KDC_Y} \| authRQ_{TU_X}$
M4. KDC_R → KDC_x: $iauthFW_{KDC_X} \| iauthAK_{KDC_Y}$
M5. KDC_x → KDC_Y: $iauthAK_{TU_X} \| iauthAK_{KDC_Y}$
M6. KDC_Y → S_Y: $SID \| iauthAK_{S_Y} \| iauthAK_{TU_X}$
M7. S_Y → TU_x: $iauthAK_{TU_X} \| CH_{S_Y} \| TKT_{S_Y}$
M8. TU_x → S_Y: $RESP_{S_Y} \| A_{TU_Y}$

M1. Assume that a user, requesting a service in a new domain Y, has a temporary user identity TU_x for its previously

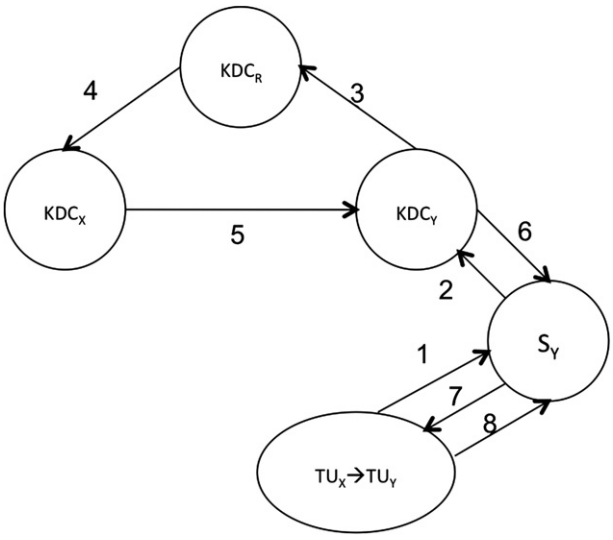

Fig. 2 – The inter-domain authentication protocol.

visited domain X. It needs to send an authentication request to the server S_Y in domain Y before accessing the desired services. The authentication request $authRQ_{TU_X}$ consists of the temporary user identity TU_x, a nonce N_{TU_X} and an encrypted message containing TU_x and N_{TU_X} using its previous temporary key K_{TU_X}.

$$authRQ_{TU_X} = TU_X \parallel N_{TU_X} \parallel \{TU_X, N_{TU_X}\}_{K_{TU_X}}$$

M2. Similarly, the server S_Y generates its authentication request $authRQ_{S_Y}$, and sends the request together with $authRQ_{TU_X}$ to its KDC_Y.

M3. Since KDC_Y cannot authenticate the visiting user, the user authentication request is forwarded to the nearest common key distribution center (KDC_R) for KDC_x and KDC_Y. The message also includes the server identity and an authentication request from KDC_Y. The server identity is used to recognize the communication session while $iauthRQ_{KDC_Y}$ is used to authenticate the common KDC_R for KDC_Y. The $iauthRQ_{KDC_Y}$ message contains KDC_Y's identity and a nonce N_{KDC_Y}, in plaintext.

M4. Upon receiving the request, KDC_R sends a forwarding message to the previously visited KDC_x. The forwarding message is encrypted using the shared key of KDC_R and KDC_x and can be represented as $iauthFW_{KDC_X} = authRQ_{TU_X}, TU_Y, K_{TU_Y}, K_{SS}\}_{K_{KDC_X}}$.

The forwarding message includes not only the authentication request issued by user, but also a new temporary principal name TU_Y, a new temporary user key and a new session key. In addition, KDC_R encrypts TU_Y, K_{TU_Y}, K_{SS} and the nonce in $iauthRQ_{KDC_Y}$ with the shared key of KDC_R and KDC_Y, and forwards it to KDC_x. This message is an authentication response to KDC_Y and can be represented as $iauthAK_{KDC_Y} = \{N_{KDC_Y}, TU_Y, K_{TU_Y}, K_{SS}\}_{K_{KDC_Y}}$.

M5. KDC_x decrypts the authentication response $iauthAK_{KDC_X}$ and gets the temporary user identity-key pair and session key. Since that KDC_x only knows the nonce and temporary user key for user TU_x, it encrypts the original nonce and new temporary user identity-key pair and session key with the pervious key K_{TU_X}. This is a message in response to the authentication request issued by user TU_x, the message is represented as $iauthAK_{TU_X} = \{N_{TU_X}, TU_Y, K_{TU_Y}, K_{SS}\}_{K_{TU_X}}$. The message is sent to KDC_Y together with the authentication response $iauthAK_{KDC_Y}$ from KDC_R.

M6. KDC_Y decrypts the authentication response $iauthAK_{KDC_Y}$, verifies the received nonce N_{KDC_Y} and extracts temporary user identity-key pair. Then, KDC_Y generates a new session identity and an authentication response to server S_Y. The response can be represented as $iauthAK_{S_Y} = \{N_{S_Y}, TU_Y, K_{SS}\}_{OTK_{S_Y}}$, where $OTK_{S_Y} = Hash(S_Y, N_{S_Y}, PW_{S_Y})$ is the one-time key of S_Y.

M7. Similar to the above description of intra-domain authentication, the server generates a challenge $CH_{S_Y} = \{S_Y, N'_{S_Y}\}_{K_{SS}}$ and a service ticket $TKT_{S_Y} = SID \parallel \{TU_Y, VT_{S_Y}, K_{SS}\}_{OTK_{S_Y}}$.

M8. The user generates a new temporary authenticator using its handover key HK_{TU_Y}. The authenticator $A_{TU_Y} = \{S_Y, VT_{TU_Y}, K_{SS}\}_{K_{TU_Y}}$ can be used for subsequent authentication. Then the user encrypts the nonce and temporary identity for the newly visited domain with the session key and sends it back as a response to CH_{S_Y}. The response is represented as $RESP_{S_Y} = \{TU_Y, N'_{S_Y}\}_{K_{SS}}$.

4. Implementation

This section discusses the integration of our protocol and the EAP framework, which we call EAP-OSNP. EAP is a client/server protocol using different authentication methods for authenticating users requesting network access. There are three entities in the EAP protocol: peer, authenticator and server. An EAP peer acts as a client requesting authentication and network services. An authenticator is the entity that controls the network access ports. An EAP server is capable of verifying users' credentials.

4.1. *EAP integration: EAP-OSNP*

Fig. 3 illustrates the EAP message flow of EAP-OSNP. An EAP peer sends *Association Request* to the EAP server. After receiving *Association Response* from the server, an EAPOL session, containing a sequence of EAP-Request/EAP-Response messages, starts for peer authentication. The *EAP-Request* and *EAP-Response* messages carry the OSNP authentication payloads, but the messages exchanged between the EAP server and KDC may optionally follow the EAP framework.

Taking intra-domain authentication as an example, the EAP peer builds an authentication request $authRQ_U$ and encapsulates it into EAP-OSNP messages. The EAP peer behaves exactly as a wireless client (U) while the EAP server as a service server (S) in OSNP. To comply with the EAP framework, one more message *EAP-Request* (type = S2U_AUTH) is required for initiating the EAP authentication. The EAP authentication succeeds when the wireless client receives the *EAP-Success* message.

To carry the OSNP payloads using EAP, we need to specify OSNP as the authentication method in the EAP packet. Fig. 4 shows the five fields in an EAP packet, including:

- *Code*: identifies the type of the EAP packet: 1 for request and 2 for response.
- *Identifier*: a sequence number used to match the request and response packets.

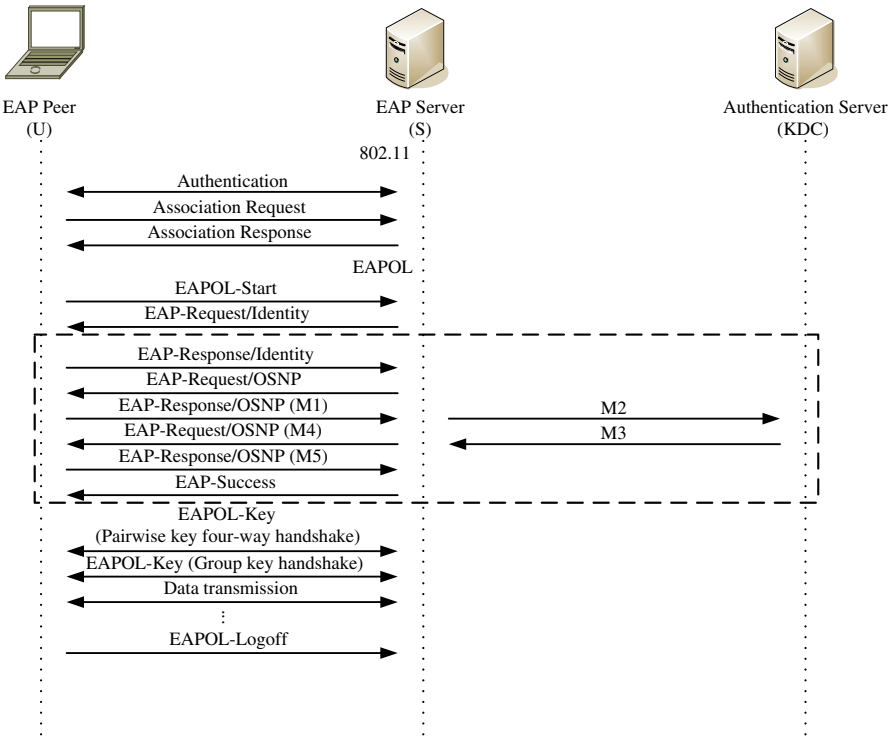

Fig. 3 – EAP-OSNP message flow: intra-domain initial authentication.

- *Length*: indicates the total length of the packet, in octets.
- *Type*: indicates the authentication method encapsulated in the EAP packet; hexadecimal 0x*DD* is reserved for EAP-OSNP in our implementation.
- *Type-Data*: contains a *Message-Type* and the *OSNP-Data*. Table 2 lists the *Message-Type* and *OSNP-Data* defined for EAP-OSNP. As mentioned in the previous paragraph, S2U_AUTH is the extra message used to initiate the EAP authentication. We also define several message types, U2S_HELLO, S2KDC_AUTH, KDC2S_AUTH, etc, for the OSNP messages M1 to M5, as described in Table 2.

Fig. 4 shows an example EAP-OSNP packet for requesting user authentication: the *Message-Type* is *U2S_HELLO* and the *OSNP-Data* is $authRQ_U$.

4.2. *EAP-OSNP implementation*

This section details the implementation of our protocol, including the authentication server, service server and wireless client.

- Authentication Server

  The authentication server (kdcd), also behaving as a key distribution center, is realized on Linux 2.6.20-21. We implement two databases to maintain user and server accounts. The first database, managing the OSNP user accounts, is separated from the Linux account database. The commands, *osnp_useradd*, *osnp_userdel* and *osnp_passwd*, are used to manage user accounts and change user passwords.

  Our protocol also requires an additional database to manage service server accounts. The server database manages the service servers registered to the authentication server. It also maintains the server identities (S) and passwords (PW_S) shared with the authentication server. In our current implementation, we support several encryption algorithms, such as IDEA, DES, AES, CAMELLIA, and hash functions, such as MD5, SHA.

- Service Server

  In our implementation, the service server supports both 128-bit and 256-bit AES encryption. The service server must register to an authentication server before it can offer services. After registration, the administrator of the service server configure its authentication server and supported cipher suites in/etc/eap.conf. For example, a service server registered to the authentication server (IP: 10.1.2.3, port:14000) is configured as:

```
osnp {
    s_identity = eapserver1
    s_passwd = gnitset
    ciphersuite = ''C_AES_128_CBC, C_AES_256_CBC''
    kdc_ip = 10.1.2.3
    kdc_port = 14000
    kdc_timeout = 4
    vt_interval = 3600
    s_port = 14000 }
```

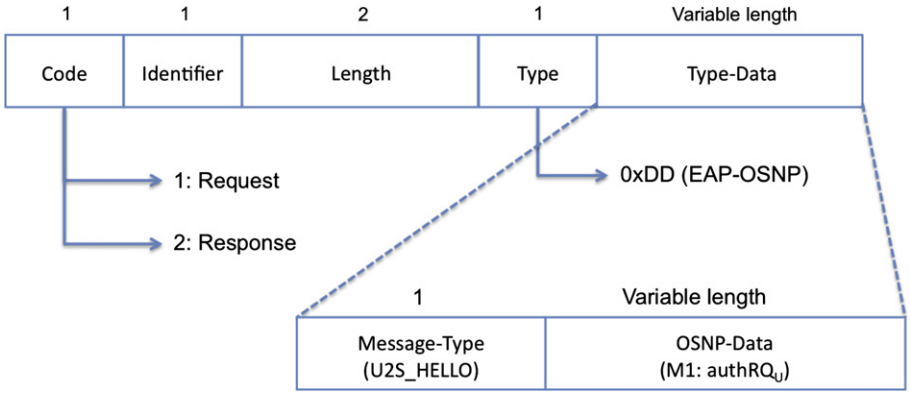

Fig. 4 – EAP-OSNP packet format: $authRQ_U$.

`vt_interval` specifies the validation time of tickets issued by the server (i.e. VT_S we mentioned in Section 3.) By running hostapd (Hostapd) (ver. 0.5.10), the service server supports and manages the wireless connections with its wireless clients.

The hostapd program is configured as a pass-though authenticator (wireless AP) and the IP address of the authentication server can be specified in the configuration file, as shown in the following:

```
ssid=wpa-osnp
ieee8021x=1
auth_server_addr=127.0.0.1
auth_server_port=1812
auth_server_shared_secret=secure
acct_server_addr=127.0.0.1
acct_server_port=1813
acct_server_shared_secret=secure
```

- Wireless Client

By revising the open source package, wpa_supplicant (Linux WPA/WPA2/IEEE 802.1X Supplicant,) (ver. 0.5.8), the OSNP wireless client can support WPA2 authentication. Similar to other OSNP components, we also need to configure the key management method, EAP method, cipher suite, ticket validation time, etc in the OSNP wireless client:

```
network={
    ssid="wpa-osnp"
    key_mgmt=WPA-EAP
    eap=OSNP
    pairwise=CCMP
    group=TKIP
    identity="eapuser"
    password="testing"
    ciphersuite="C_NULL, C_AES_256_CBC"
    vt_interval=4800
    priority=20 }
```

In this example, the wireless client uses WPA-EAP (EAP-OSNP) to connect to the access point with SSID "wpa-osnp". Its cipher suite is configured as 256-bit AES, while the validation time (VT_U) is 4800 seconds.

Table 2 – EAP-OSNP messages.

Message-type	OSNP-data
S2U_HELLO	S
U2S_HELLO	M1: $authRQ_U$
S2KDC_AUTH	M2: $authRQ_S \| authRQ_U$
KDC2S_AUTH	M3: $SID \| authAK_S \| authAK_U$
S2U_AUTH	M4: $authAK_U \| CH_S \| TKT_S$
U2S_AUTH	M5: $RESP_S \| A_U$

4.3. *Experimentation*

We validate the proposed protocol on the SWOON (Secure Wireless Overlay Observation Network) testbed (Huang et al., 2008). In this section, we brief the SWOON testbed and explain how we test and verify our protocol on SWOON. The experiments are built on Linux kernel version 2.6.20-21 with kernel cryptographic API enabled. In the end of this section, we compare our protocol with other EAP methods in terms of the protocol handshaking performance.

4.3.1. *SWOON testbed*

Taking Emulab (White et al., 2002; Emulab) and DETER (Benzel et al., 2006; cyber-DEfense Technology) as its basis, the SWOON testbed is an emulation testbed for wireless networks. The SWOON testbed is a comprehensive and flexible testbed allowing designers to test and validate their new wireless protocols and systems without building a physical test environment. The SWOON testbed supports heterogeneous wireless networking technologies, including WiFi and WiMAX. On SWOON, we can specify the desired wireless network topology for experimentation.

SWOON uses virtual wireless devices which encapsulate wireless packets with UDP broadcast headers, so that we can send/receive wireless packets over the DETER testbed (an emulated ethernet testbed). The SWOON testbed can emulate wireless broadcast, delivery latency and packet loss. Such features help us to validate correctness and measure performance under various network conditions.

4.3.2. *Setting up experiments*

We setup an experiment for measuring the performance of the intra-domain authentication protocols of various EAP

methods, including EAP-OSNP, EAP-TLS, EAP-TTLS/MD5 and PEAPv0(MS-CHAPv2). Fig. 5 illustrates the wireless network topology of our experiment, in which there are five nodes: one authentication server (*kdc*), two service servers (*s0*, *s1*), one wireless station (*sta*), and one wired node (*dst*).

The authentication server, *kdc*, and the two service servers are located in the same domain and thus connected via a private LAN switch (*plan*). The service servers, *s0* and *s1*, behaving as wireless APs, are both equipped with two network interfaces: one wired network interfaces card (NIC) and one wireless NIC. The wireless station (*sta*) is equipped with only one wireless NIC. Since these three nodes are radio reachable, we connect them to the same switch (*wireless*). The wired node (*dst*), offering public services, is located outside the public network. It can be reached via the external switch (*lan*).

For the intra-domain authentication, the wireless station *sta*, requesting the service running on *s0*, is authenticated by *kdc*. To request services running on other server, says *s1*, *sta* executes the subsequent authentication protocol to obtain a new session key without involving *kdc*. If the wireless station *sta* wants to request for services running on nodes in another domain, *dst*, for example, the handover authentication is proceeded. With the above scenarios, we can measure and compare EAP methods in terms of the performance of user authentication.

4.3.3. *Experiment results*

By running a packet sniffer tool, wireshark (Wireshark) (ver. 1.0.0) on *sta*, we capture the authentication messages exchanged between the service server and the wireless station. After analyzing the captured messages, we can calculate the time duration (Δt) between *EAP-Response/EAP-Identity* and *EAP-Success* shown in Fig. 3.

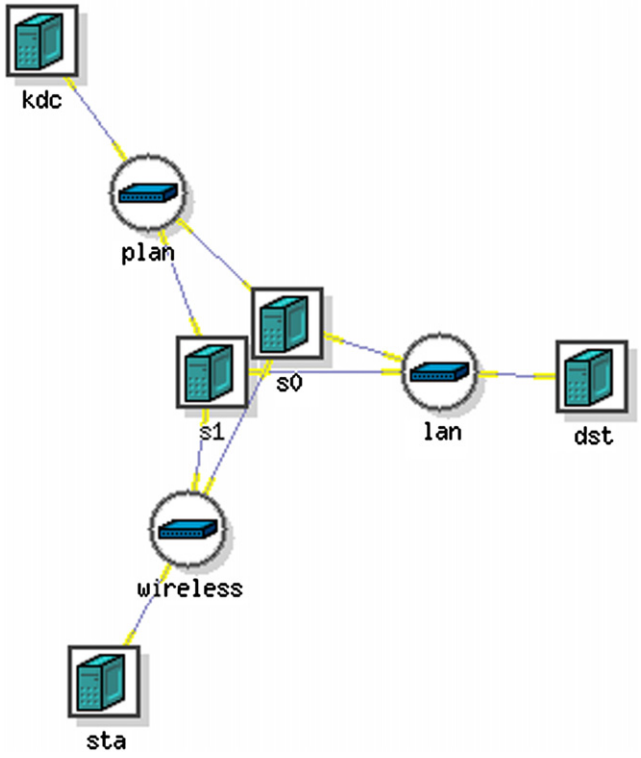

Fig. 5 – The topology of our experiment.

We measure the time cost by running 1000 user authentications on each protocol and obtain the average cost shown in Table 3. The table shows the mean value for the average time cost of authentication. Compared with TLS, TLS/MD5 and PEAPv0(MSCHAPv2), OSNP takes 10.37 ms for an initial authentication, 4.43 ms for a subsequent authentication and 10.68 ms for a handover authentication.

From the result, we conclude that EAP-OSNP has a better performance than any other EAP method. Even the curves of EAP-OSNP initial authentication and handover authentication are quite close, we still reduce the computational loads of KDC since no KDC is involved during handover.

5. Analysis and comparisons

This section discusses how our protocol can protect against the attacks. We also compare the OSNP protocol with other EAP-based protocols in terms of costs (communication, storage, etc) and performance.

5.1. *Security analysis*

- Trivial Substitutions and Replay Attack

  Since all of the proposed protocols are nonce-based and every credential and ticket in our protocols contains the nonces used to verify the freshness of that credential, trivial substitutions and replays attacks can be easily detected. Similar to other nonce-based protocols, the challenger starts a timer and waits for a response. If the timer expires before receiving the response, the challenger assumes that the message is either lost or corrupted and must issue a new challenge.

- Man-in-the-Middle Attack

  Since all critical messages in our protocol are encrypted to prevent eavesdropping, it is nearly impossible to modify the messages exchanged between entities. However, if an attacker *A* eavesdrops the communication channel between *U* and *S*, he can replace the authentication request $authRQ_U$ with $authRQ_A$. The replaced $authRQ_A$ is forwarded to the *KDC* together with $authRQ_S$. The attacker may be successfully authenticated by the *KDC* if he is a legitimate user in the system, but the Man-in-the-Middle attack still fails because the attacker cannot generate a correct $authAK_U$ to respond to the $authRQ_U$. Therefore, we conclude that a Man-in-the-Middle attack would not succeed against the OSNP protocol.

Table 3 – Average time costs of EAP methods.

	OSNP			TLS	TTLS/MD5	PEAPv0
	Initial	Subsequent	Handover			
Δt (ms)	10.37	4.43	10.68	69.47	59.32	66.24
Messages	4	4	4	14	12	18

Table 4 – Expected time cost of compromising OTKs.

Length of PW_U	MD5	SHA1	SHA512
3	1.140 (s)	1.325 (s)	2.041 (s)
4	110.967 (s)	132.108 (s)	194.784 (s)
6	175.698 (min)	209.171 (min)	308.408 (min)
8	1101.161 (day)	1310.950 (day)	1932.904 (day)

The experiment is conducted on Ubuntu 8.4.2 (kernel 2.6.24) with Intel Pentium D CPU 2.8 GHz and memory size of 1 GB. The guessing space is limited to 95 printable characters (0x20 to 0x7E).

- User Impersonation Attack with Compromised Session Keys

  Since each session key is used only for a single authentication session and is discarded after authentication, an impersonation attack with a compromised session key can be prevented. In our authentication protocols, we do not rely on timestamps or temporary keys. Taking intra-domain initial authentication as an example, this kind of attack can be easily detected by a server in M3 by checking the freshness of the nonce in $authAK_S$. If the intruder substitutes N_S in M2 and replays M3, the server can still detect that M3 is simply a forged message by verifying the nonce in $authAK_S$. The intruder will be rejected even if he holds a compromised session key.

- Forward Secrecy

  Our protocol addresses forward secrecy. The disclosure of long-term secret keying material used to derive an agreed key does not compromise the secrecy of agreed keys from earlier runs (Ohba et al., 2007). In our protocol, keys are chosen randomly, and the one-time key itself is used as a key which changes with each use.

- 802.1X Identity Privacy

  When an eavesdropper is listening on network traffic, the authentication process exposes the identity of the EAP peer. Even with a stolen identity, the eavesdropper still cannot login into the system without the correct one-time key. Taking intra-domain initial authentication as an example, suppose the eavesdropper stores the user's identity from previous sessions. It could then generate a forged authentication request $authRQ_U$. However, the request would fail authentication.

Table 5 – Authentication and security analysis.

EAP methods	TLS	OTP	Kerberos	OSNP
Server authentication	CERT	N/A	PW	OTK
Client authentication	CERT	OTK	PW	OTK
Mutual authentication	No	No	Yes	Yes
Replay attack	Yes	Yes	Yes	Yes
Dictionary attack	Yes	No	No	No
Brute-force attack	No	Yes	No	Yes
Identity privacy protection	No	No	No	No
Man-in-the-middle attack	Yes	No	Yes	Yes
User Impersonation attack	No	No	Yes	Yes
Forward secrecy	No	Yes	Yes	Yes

CERT, Certificate; PW, Password; OTK, One-time key.

Table 6 – Performance comparison: computation costs.

Authentication	Operations	Kerberos				OSNP			
		KDC	S_{old}	S	U	KDC	S_{old}	S	U
Initial	Random	2	–	0	2	2	–	2	1
	Hash	4	–	3	4	2	–	1	1
	En/decrypt	6	–	5	5	5	–	5	5
Subsequent	Random	–	–	–	–	–	–	2	1
	Hash	–	–	–	–	–	–	1	0
	En/decrypt	–	–	–	–	–	–	4	4
Handover	Random	–	–	–	–	–	0	3	1
	Hash	–	–	–	–	–	1	1	0
	En/decrypt	–	–	–	–	–	3	5	5

- Dictionary Attack

  Since the user's identity and nonce are sent in plaintext, they may suffer from the brute-force attack. An attacker may guess PW_U by dictionary guessing and comparing the hash values of (U, N_U, *guess*) and OTK_U. We made a small experiment to obtain the time cost of compromising a PW_U, as listed in Table 4. From the experiment results, we conclude that the passwords should be restricted to lengths longer than 8 *bytes* and the frequency of changing the password can be determined according to the selected hash function.

Table 5 summarizes the analysis of some EAP methods, including EAP-TLS, EAP-OTP, EAP-Kerberos and EAP-OSNP. We compare their characteristics and capabilities against attacks of these methods in the table.

5.2. *Performance analysis*

Tables 6 and 7 show the performance of Kerberos and OSNP, in terms of computation, communication and storage. Table 7 shows the number of messages for mutual authentication and the number of messages submitted by a user. OSNP requires a constant number of messages independent of the number of KDCs between the user's visited domain and home domain. This reduces the time required for roaming from one domain to another. Compared with the Kerberos protocol, OSNP requires only two messages on the user side. This is feasible and practical for mobile networks with low data rates and bandwidth. It is also good for battery-powered mobile devices.

Table 7 also compares the number of shared keys among these protocols. Consider a hierarchy with N domains. The

Table 7 – Performance comparison: communication and storage costs.

	#Msg for mutual authentication	#Msg initiated from user	Type of trust	#Shared keys	Mobility support
Kerberos	$2m+4$	$m+2$	P&H	$O(N)$	No
OSNP	8	2	P&H	$O(N)$	Yes

m, Number of KDCs between the user's visited domain and home domain; N, Number of domains; P&H, Peer and Hierarchical.

number of shared keys in OSNP is proportional to the number of domains, which is the same as the number of shared keys in Kerberos V5.

6. Conclusion

In this paper, we integrated the one-time key with our nonce-based authentication protocol, which efficiently supports initial, subsequent and handover authentication. Our protocol requires five messages for initial authentication; three for subsequent authentication and five for handover authentication. Although five messages are required for handover authentication, no KDC is involved in authenticating the roaming user. Then, we extended the intra-domain authentication protocol to an inter-domain authentication protocol, which requires eight messages for mutual authentication, regardless of the number of hops between the visited and home domains. In all our authentication protocols, only two messages are sent by the user. Such a design is feasible for a mobile network with limited bandwidth and for those battery-powered mobile devices.

Since KDCs are transparent to users in OSNP, only registered servers can communicate with KDCs directly. This architecture is suitable for modern mobile networks, where mobile devices only need to connect to a visited server for authentication, without knowing the location of the KDC.

Acknowledgement

This effort was partially supported by the International Collaboration for Advancing Security Technology (iCAST) and Taiwan Information Security Center (TWISC) projects, sponsored by National Science Council under the grants NSC96-3114-P-001-002, NSC97-2745-P-001-001, NSC96-2219-E-009-013 and NSC97-2219-E-009-006.

Appendix.

In the appendix, we use BAN logic (Burrows et al., 1989) and its enhancement (Shieh et al., 1999) to explain why our protocol can reach the goals of mutual authentication for initial and subsequent authentication of intra-domain authentication. The intra-domain handover is a derivation of initial authentication, replacing KDC with the previous server and the inter-domain authentication is an extension of the initial authentication; the security of these two types of authentication is guaranteed by that of the initial authentication.

The BAN logic states that the mutual authentication is complete between two parties A and B, if there is a K such that

A believes $A \overset{K}{\leftrightarrow} B$,
B believes $A \overset{K}{\leftrightarrow} B$,
A believes B believes $A \overset{K}{\leftrightarrow} B$,
B believes A believes $A \overset{K}{\leftrightarrow} B$.

Initial authentication

The objectives of the initial authentication are to prove: the presence of both parties to each other, and the receipt of a ticket and a session key at the user side. Assume that

$$U \text{ believes } U \overset{OTK_U}{\longleftrightarrow} \text{KDC, and} \quad (1)$$

$$S \text{ believes } S \overset{OTK_S}{\longleftrightarrow} \text{KDC.} \quad (2)$$

The proof is given in two parts: to authenticate S by U, and to authenticate U by S.

For the first part, since U receives $authAK_U$, CH_S and TKT_S in M4, he can decrypt $authAK_C$ and get the session key K_{ss}. By applying annotation rule and formula 1, we obtain

U believes $U \overset{OTK_U}{\longleftrightarrow}$ KDC,
U sees $\{N_U, S, U \overset{K_{SS}}{\longleftrightarrow} S\}OTK_U$, and
U believes KDC said$(N_U, S, U \overset{K_{SS}}{\leftrightarrow} S)$.

Since that N_U is generated by U, we have the following hypothesis:

U believes fresh $(N_U, S, U \overset{K_{SS}}{\longleftrightarrow} S)$.

The nonce-verification rule applies and yields

U believes $U \leftrightarrow S$.

By decrypting the CH_S, U verifies the server identity. Similarly, we obtain

U sees $\{S, N'_S\}K_{ss}$,
U believes S said $(S, N'_S, U \overset{K_{SS}}{\longleftrightarrow} S)$,
U believes S believes $U \overset{K_{SS}}{\longleftrightarrow} S$.

The second part is proved by M3 and M5. S receives $authAK_S$ in M3, and he can decrypt the token and extract the session key K_{ss}. Then, S decrypts $RESP_S$ using K_{ss} to get N'_S. Similarly, applying the annotation, the message-meaning, jurisdiction rules, and formula 2, we obtain

S believes $S \overset{OTK_S}{\leftrightarrow}$ KDC,
S sees $\{N_S, U, U \overset{K_{SS}}{\longleftrightarrow} S\}OTK_S$,
and S believes KDC said $(N_S, U, U \leftrightarrow S)$.Since that N_S is generated by S, we have the following hypothesis:
S believes fresh $(N_S, U, U \leftrightarrow S)$, and S believes $U \leftrightarrow S$.

Similarly, N'_S is generated by S, we apply the nonce-verification rule and jurisdiction rule and obtain

S sees $\{N'_S\}K_{ss}$,
S believes fresh $(N'_S, U \overset{K_{SS}}{\longleftrightarrow} S)$,
S believes U said $(N'_S, U \overset{K_{SS}}{\longleftrightarrow} S)$,
and S believes U believes $U \overset{K_{SS}}{\longleftrightarrow} S$.

It proves that our initial authentication can achieve the following goals at the end of the authentication round:
U believes $U \overset{K_{SS}}{\leftrightarrow} S$,
S believes $U \overset{K_{SS}}{\longleftrightarrow} S$,

U believes S believes $U \overset{K_{SS}}{\longleftrightarrow} S$,

and$S$ believes U believes $U \overset{K_{SS}}{\longleftrightarrow} S$

Subsequent authentication

In our subsequent authentication, S decrypts TKT_S and extracts U, VT_S and K_{ss}. After checking the validation of VT_S, K_{ss} is still validate. We can apply the above formal rules, and obtain

U believes $U \overset{K_{SS}}{\longleftrightarrow} S$,

U sees $\left\{N_U, U \overset{K'_{SS}}{\longleftrightarrow} S\right\}K_{ss}$,

U believes S said $\left(N_U, U \overset{K'_{SS}}{\longleftrightarrow} S\right)$,

U believes fresh $\left(N_U, U, U \overset{K'_{SS}}{\leftrightarrow} S\right)$, and

U believes S believes $U \overset{K'_{SS}}{\longleftrightarrow} S$.

For S, we also prove that

S believes $S \overset{OTK_S}{\longleftrightarrow}$ KDC,

S sees $\{U, VT_S, U \overset{K_{SS}}{\leftrightarrow} S\}OTK_S$,

S believes KDC said $(U, VT_S, U \overset{K_{SS}}{\longleftrightarrow} S)$.

So,

S believes $U \overset{K_{SS}}{\leftrightarrow} S$.

After receiving $RESP_S$, which contains a nonce N_S and a new session key K'_{ss} generated by S, we obtain

S sees $\{N_S\}K'_{ss}$,

S believes fresh $\left(N_S, U \overset{K'_{SS}}{\longleftrightarrow} S\right)$,

S believes U said $\left(N_S, U \overset{K'_{SS}}{\longleftrightarrow} S\right)$,

S believes U believes $U \overset{K'_{SS}}{\longleftrightarrow} S$.

This proves that our subsequent authentication can achieve the above goals.

REFERENCES

Aboba B, Blunk L, Vollbrecht J, Carlson J, Levkowetz H. Extensible Authentication Protocol (EAP). RFC 3748; June 2004.

Baek K-H, Smith SW, Kotz D. A survey of WPA and 802.11i RSN authentication protocols. Tech. rep., Dartmouth College Computer Science, TR2004-524; November 2004.

Benzel T, Braden R, Kim D, Neuman C, Joseph A, Sklower K, Ostrenga R, Schwab S. Experience with DETER: a testbed for security research. Testbeds and research infrastructures for the development of networks and communities, 2006. TRIDENTCOM 2006. 2nd International Conference on, 10p; March 2006.

Burrows M, Abadi M, Needham R. A logic of authentication. In: SOSP'89: proceedings of the Twelfth ACM Symposium on operating systems principles. New York, NY, USA: ACM; 1989. p. 1–13.

Chien H-Y, Jan J-K. A hybrid authentication protocol for large mobile network. Journal of Systems and Software 2003;67:123–30.

cyber-DEfense technology experimental research laboratory testbed. http://www.isi.edu/deter/.

Emulab – Network emulation testbed. http://www.emulab.net/.

Hostapd: IEEE 802.11 AP, IEEE 802.1X/WPA/WPA2/EAP/RADIUS authenticator. http://hostap.epitest.fi/hostapd/.

Huang YL, Tygar JD, Lin HY, Yeh LY, Tsai HY, Sklower K, Shieh SP, Wu CC, Lu PH, Chien SY, Lin ZS, Hsu LW, Hsu CW, Hsu CT, Wu YC, Leong MS. SWOON: a testbed for secure wireless overlay networks. In: CSET' 08; 2008.

Hwang R-J, Su F-F. A new efficient authentication protocol for mobile networks. Computer Standards & Interfaces 2005;28: 241–52.

Ieee. (IEEE 802.1X), wireless LAN media access local and metropolitan area networks: port-based network access control. Tech. rep., IEEE; June 2001.

Ieee. IEEE 802.11i/D4.0: Medium Access Control (MAC) Security Enhancements. Tech. rep.; May 2003.

Liang W, Wang W. A lightweight authentication protocol with local security association control in mobile networks. In: IEEE military communications conference (MILCOM) 2004, vol. 1; 2004, pp. 225–31.

Linux WPA/WPA2/IEEE 802.1X Supplicant. http://hostap.epitest.fi/wpa_supplicant/.

Macnally C. Cisco LEAP protocol description, http://lists.cistron.nl/pipermail/cistron-radius/2001-September/002042.html; 2001.

Neuman C, Yu T, Hartman S, Raeburn K. The Kerberos Network Authentication Service (V5). RFC 4120; July 2005.

Ohba Y, Das S, Dutta A. Kerberized handover keying: a media-independent handover key management architecture. In: MobiArchÕ07, Kyoto, Japan; August 2007.

Ppp Eap TLS Authentication Protocol. RFC 2716, October 1999.

Remote Authentication Dial In User Service (RADIUS). RFC 2865, June 2000.

Shieh S, Ho F, Huang Y. An efficient authentication protocol for mobile networks. Journal of Information Science and Engineering 1999;15(15):505–20.

The flexible authentication via secure tunneling extensible authentication protocol method EAP-FAST. RFC 4851, May 2007.

Wireshark: network protocol analyzer. http://www.wireshark.org/.

White B, Lepreau J, Stoller L, Ricci R, Guruprasad S, Newbold M, et al. An integrated experimental environment for distributed systems and networks. SIGOPS Operative Systematic Review 2002;36(SI):255–70.

Xiao-rong C, Qi-yuan F, Chao D, Ming-quan Z. Research and realization of authentication technique based on OTP and kerberos. In: Eighth international conference on high-performance computing in Asia–Pacific Region (HPCASIA'05); 2005, pp. 409–16.

Zrelli S, Shinoda Y. Specifying Kerberos over EAP: towards an integrated network access and Kerberos single sign-on process. In: Advanced information networking and applications. AINA'07, 21st international conference; 2007, pp. 490–97.

Y. L. Huang is an assistant professor of Electrical and Control Engineering at National Chiao-Tung University, Taiwan. She received her BS and PhD degrees in Computer Science and Information Engineering from National Chiao-Tung University, Taiwan in 1995 and 2001, respectively. She is a member of Phi Tau Phi Society since 1995. Her research interests include network security, wireless security, secure testbeds, secure control systems, VoIP, embedded systems, etc.

P. H. Lu received his BS degree in Electrical and Control Engineering from the National Chiao-Tung University, Taiwan

in 2008. His research interests include network security, network administration and wireless testbeds.

J. D. Tygar is Professor of Computer Science at UC Berkeley and also a Professor of Information Management at UC Berkeley. He works in the areas of computer security, privacy, and electronic commerce. His current research includes privacy, security issues in sensor webs, digital rights management, and usable computer security. His awards include a National Science Foundation Presidential Young Investigator Award, an Okawa Foundation Fellowship, a teaching award from Carnegie Mellon, and invited keynote addresses at PODC, PODS, VLDB, and many other conferences. Before coming to UC Berkeley, Dr. Tygar was tenured faculty at Carnegie Mellon's Computer Science Department, where he continues to hold an Adjunct Professor position. He received his doctorate from Harvard and his undergraduate degree from Berkeley.

A. D. Joseph is now an associate professor at the University of California, Berkeley. He is a former member of Professor Frans Kaashoek's research group Parallel and Distributed Operating Systems in MIT's Laboratory for Computer Science. His research interests include operating systems and application issues related to wireless mobile computing and parallel and distributed systems.