Network-based AAA Routing over Heterogeneous Wireless Networks

Jun-Cheol Park[†] **and** **Sung Ik Jun**[††]

Department of Computer Engineering
Hongik University, Seoul 121-791, Korea

Information Security Research Division
ETRI, Daejeon 305-700, Korea

Summary
This paper addresses the problem of AAA routing in roaming situations over heterogeneous wireless networks. While traveling, a mobile user often initiates an authentication process with the user's home network AAA server to use the resources of a remote network. In the process, the user wants to affect the selection of a roaming path to the user's home network, because pricing and other services differ by the path. We present a network-based solution for the selection that eliminates the necessity of user's inquiring, collecting, and analyzing data over, possibly, heterogeneous wireless networks. The solution assists the user in selecting a roaming path with respect to a user-preferred metric and performing a mutual authentication afterward via the selected path. We first propose a routing and label computation algorithm for generating pre-provisioned routes through which authentication traffic would pass. At the heart of the solution is a mutual authentication protocol that, together with registration, handshake, and route deviation detection schemes, provides a secure, privacy preserving, and ultra-lightweight way of authentication. Using password-based identification, the solution is generic and requires no particular transmission mechanism, payload, or routing framework. It should be incrementally deployable over selected AAA servers interoperable with existing AAA infrastructure not knowing of the solution.

Key words: *AAA routing, authentication, security, heterogeneous wireless networks*

1. Introduction

In roaming situations, a mobile user who wishes to use the resources of a remote network needs to be authenticated first before the user receives desired service. AAA routing figures out which mediating networks are available and lets user choose which network to pass and which identity to authenticate. As the user is charged according to the usage of mediating networks, it is necessary for the user to select which route to take to the home network AAA server. The user may prefer one path to another for pricing, QoS, or other reasons. But, the user does not know if a nearby access network has a direct or indirect roaming relationship with the network he is attempting to authenticate with. Hence the availability and other information on roaming paths must be somehow accessible to the user. Using the information, the user is able to choose an appropriate authentication credential to use too.

The AAA routing has been studied in the context of network discovery and selection problem [1,2,3,4,5]. The network discovery and selection is on the discovery of available networks followed by a selection of one of them by a mobile user in roaming. It becomes relevant when there are several access networks belonging to different operators, the user has multiple sets of authentication credentials, or there are various ways of providing roaming with different service parameters.

Mechanisms for AAA routing are either network-based or user-based. In user-based mechanisms, user (terminal) needs to query each of the available networks to collect data useful for the decision. Thus such a mechanism can place a significant overhead on the wireless links. Even if data are available, the network operators might not want to directly give the information to the user who is yet to be authenticated. The mechanisms also require a single terminal to deal with heterogeneous wireless networks. In network-based mechanisms, however, networks perform data collection and analysis on behalf of users. A user makes a final decision based on the provided information. Network-based mechanisms are more bandwidth efficient since they would significantly reduce the necessity of users being involved in data gathering and analysis. Accordingly, network-based approaches are preferred over user-based approaches when sufficient roaming paths are available over heterogeneous wireless networks.

In this paper, we propose a secure, lightweight, and generic solution for the AAA routing problem. It ensures a user's authentication traffic to be securely routed to/from the user's home network AAA server via a pre-provisioned route upon the user's approval. It ensures user privacy by using a randomly selected one-time identifier

Manuscript received December 5, 2008
Manuscript revised December 20, 2008

that changes each authentication request of a user. In addition, it provides a way for the home network AAA server to verify that the path of mediating networks used was the one the user has requested.

The rest of this paper is as follows. The next section proposes a graph-based model for representing a group of provider networks with roaming agreements. In section 3, we present an algorithm for computing routes and labels in a roaming group graph that generates pre-provisioned paths for roaming users. Then, in section 4, for the AAA routing problem, we give a novel solution including a registration protocol, a handshake protocol with an access network, a mutual authentication protocol, and a feature for route deviation detection. Security analysis of the proposed solution is given in section 5. We present related work in section 6. Conclusion is given in section 7.

2. Graph Model

For the presentation of the solution, we use a graph model where a directed weighted graph represents a roaming group with agreements. In a graph, a node means a provider network and an edge $\langle A, B\rangle$ means A offers some service to B according to the roaming agreement between A and B. For a bilateral agreement between A and B, a pair of edges $\langle A, B\rangle$ and $\langle B, A\rangle$ must exist, where the contracted terms may not be symmetric. An edge $\langle A, B\rangle$ is associated with a positive weight, which represents the cost for the service provided to B by A based on a metric. A roaming group is formed by a set of provider networks that are related directly (i.e., connected by a single edge) or indirectly. By forming a roaming group, it is feasible for a provider network to use the resources of other networks that are not necessarily direct roaming partners of the network. For example, a user of network A can use network C with the help of an intermediate network B, even though A has no direct roaming agreement with C.

To evaluate roaming paths in terms of various service qualities, one needs to define a set of common metrics leveraging different network characteristics of numerous heterogeneous networks. For example, one may use only bandwidth and price as metrics and classify services based on the offered bandwidth per unit price. Then the edge weights in a roaming group graph would be set by the offered services judged by the metrics. For example, if A has agreed to provide 5Mbps to B, and C to provide 10Mbps to D for the same price, then one might assign the edge weight of $\langle A, B\rangle$ to 10 and the edge weight of $\langle C, D\rangle$ to 5. If a provider network charges each customer a fixed rate per month, the provider can compute its offered bandwidth per unit price by dividing the average amount of monthly traffic per customer by its fixed rate. A more complex scheme for weight assignment is feasible that may incorporate other QoS characteristics, for example, packet loss probability and propagation delay. In addition, edge weight may be reflecting whether or not a certain minimum service level is guaranteed. These sophisticated metrics [6,4] are beyond the scope of this paper. At any rate, if a metric, either simple or complex, is defined, it is feasible to construct a graph with edge weights assigned according to the metric. Note that each provider network can still define and use its own metric for charging as long as the provider, if asked, is willing to tell its service level according to a different metric. A weighted graph constructed in this way is unique and used only for computing routes to the node (provider network) that defined the metric.

Each provider network is assumed to know how to route packets to its roaming partners (to its AAA server, in particular), which is feasible via a routing protocol or even a manual configuration of routes. Packet transmission and reception among provider networks are actually done by the AAA servers or proxies of the networks. When making a roaming contract, two participant networks are assumed to securely exchange a secret key via a secure channel. With the key, the networks with a roaming agreement can provide message confidentiality via symmetric encryption.

3. Algorithm for Computing Routes and Labels in a Roaming Group Graph

The algorithm is destination-initiated, which means any provider network can initiate computing routes back to it. This section assumes that a given roaming group is represented by a directed weighted graph using a metric specified by a particular destination network. The algorithm relies on message flooding in a roaming group graph. An initial routing message is generated at a destination node in the graph. On the receipt of a routing message, each node updates its current best path to the destination based on the information carried by the message, modifies the message, and then floods the modified message to all neighbors with edges toward the node. Each destination fully controls its route computation time, which is uncorrelated with the route computation of other destination. Moreover our routing is independent of other routing scheme, if any, used to compute paths for

other traffic, for example, non-AAA traffic.

Routing packet in the algorithm has the form [*destination, label, cost, seq_no, hop_count*], where *label*, *cost*, and *hop_count* values are changed hop-by-hop basis. *destination* is the originating node that initiated the routing packet. *label* is for recognizing the path that the packet traversed so far, and *cost* is the cost (sum of edge weights) of the path. *seq_no* is an increasing sequence number given by the originating node to tell the currency of the packet. *hop_count* specifies the range of the packet forwarding and thus allows the originating node to limit the length of possible roaming paths from it. The *hop_count* value decrements by 1 at each relaying node and any node that makes a packet's *hop_count* 0 must discard the packet immediately.

Each node maintains a routing table and a set of secret values. A routing table consists of entries, where an entry is a $\langle$ *destination, label, cost, seq_no next_hop* $\rangle$ tuple. *next_hop* is the neighbor that sent the packet. Other terms are explained above. A path with a smaller cost will replace an existing entry representing a path to the same destination. In addition, each node randomly selects and maintains a set of secret values, one for each incoming edge to the node. A secret value for an edge is to be exclusive-ORed with the label of a packet before the packet is forwarded to the other end of the edge. For confidentiality, each packet is encrypted with a symmetric secret key before being sent to a neighbor of the node. The key used is the one shared by the corresponding neighbor.

A node a performs the below algorithm to process a packet $[z,l,c,n,h]$ received from a neighbor b. The algorithm updates an entry of a's routing table if a better route to the destination z is found. Throughout the paper, we use XOR or the symbol $\oplus$ for the bitwise exclusive-OR operation.

Figure 2 shows how the algorithm computes routes and corresponding labels to a particular destination in a roaming group graph. Note that only edges directed toward node *A* are depicted and the common destination *A* and sequence number *n* in the packets are omitted in the figure to reduce clutter.

Suppose node *A* initiates the algorithm by issuing a routing packet to each of its neighbors. *A* transmits packets $[A, s_0, 0, n, h]$ and $[A, s_0', 0, n, h]$ to *B* and *C*, respectively, where h is the hop limit within which the packet is allowed to travel. In the example, h is set to 3. Packets leaving *A* would eventually arrive *E* via one of its neighbors *D*, *B*, *C*, and *F*. There are 4 paths from *E* to *A* : (path1) *E-D-B-A*, (path2) *E-B-A*, (path3) *E-C-A*, and (path4*) E-F-C-A*.

Algorithm ==========

$c' \leftarrow c +$ cost $\langle a,b \rangle$; // cost $\langle a,b \rangle$ is the weight of // the edge $\langle a,b \rangle$

look up the value z in the destination fields of a's routing table

if a matching entry $\langle z, \overline{l}, \overline{c}, \overline{n}, d_j \rangle$ found

  if $(n \geq \overline{n})$ // the packet is a recent one?

    if $(c' < \overline{c})$ // with a smaller cost?

      replace $\langle z, \overline{l}, \overline{c}, \overline{n}, d_j \rangle$ with the new one $\langle z, l, c', n, b \rangle$;

    // $c' \geq \overline{c}$: goto the hop count check

  else // $n < \overline{n}$

    discard the packet and **stop**;

else // no matching entry

  create and insert an entry $\langle z, l, c', n, b \rangle$ into the table;

// hop count check

$h' \leftarrow h-1$;

if $(h' > 0)$ // hop count not expired?

  for each edge $\langle d_i, a \rangle$, $1 \leq i \leq k$, and $d_i \neq b$

    transmit $[z, l \oplus s_i, c', n, h']$ to d_i;

  // s_i is the secret value for $\langle d_i, a \rangle$ selected by a

==========

Fig. 1 Route Update and Forwarding at a on Receipt of $[z,l,c,n,h]$ from b

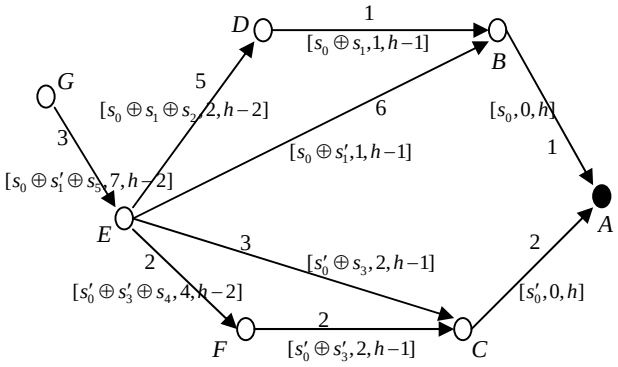


Fig. 2 An Example for Route and Label Computation (Destination: *A*)

The packet $[A, s_0 \oplus s_1 \oplus s_2, 2, n, h-2]$, when arrived at *E* via the path1 in reverse order, would not be forwarded further since the hop count makes 0. From the packet, *E*

gets a path of cost 7(2+cost$\langle E, D\rangle$) to the destination *A*, which corresponds to the path1. Similarly, the packet $[A, s_0 \oplus s_1', 1, n, h-1]$ traveled along the path2 in reverse order would notify *E* of a path of cost 7(1+cost$\langle E, B\rangle$) to the destination *A*. Since these two paths are of equal cost, the one that arrived at *E* later would be disregarded. The packet $[A, s_0' \oplus s_3, 2, n, h-1]$ from the path3 would generate a path of cost 5(2+cost$\langle E, C\rangle$) that would replace the existing path, if any, since its cost is smaller. The path4 with cost 6(4+cost$\langle E, F\rangle$) would be also recognized by the packet $[A, s_0' \oplus s_3' \oplus s_4, 4, n, h-2]$, but be defeated by the path3. The arrival order of the packets from the 4 paths is unpredictable. Regardless of the real arrival order, however, the path to survive should be always the path3, which has the smallest cost. This path will be stored in *E*'s routing table as follows.

Table 1: *E*'s Routing Table Entry for Destination *A*

destination	label	cost	seq number	next hop
A	$s_0' \oplus s_3$	5	*n*	*C*

To deliver this path information to *E*, *C* must contain an entry in its routing table as below.

Table 2: *C*'s Routing Table Entry for Destination *A*

destination	label	cost	seq number	next hop
A	s_0'	2	*n*	*A*

A packet arrived at *E* will be updated and be sent to its neighbors as long as the packet's decreased hop limit is still greater than 0. For example, the packet $[A, s_0 \oplus s_1', 1, n, h-1]$ via the path2 computes its h' as 1, and be updated and transmitted to *G* as shown in Figure 2. For a certain destination enforcing a limited hop count, a remote node does not necessarily compute a shortest path to the destination. For example, suppose a new node *H* and edges $\langle E, H\rangle$ and $\langle H, D\rangle$ are added into the graph, where cost$\langle E, H\rangle$ =cost$\langle H, D\rangle$ =1. Then the cost of the path *E-H-D-B-A* will be 4, which is less than the cost of the path3. However, if *A* sets the hop limit 3 initially, not every node would compute *the* shortest path to the destination *A*. The packet issued by *A* and relayed over the path *E-H-D-B-A* in reverse order would make its hop count 0 at *H*, where it would be discarded. As a result, *E* cannot receive the packet specifying the path passing through *H* and records the path *E-C-A* as the shortest path to *A*. On the other hand, the nodes *H*, *D* and *B* on the shortest path *H-D-B-A* compute the shortest path to *A* correctly since they are within the hop count range. This inconsistency is no error and is, in fact, intended by the initiator *A*, who wanted to disallow traffic coming from a node over 3 hops away.

4. Secure and Lightweight Solution for AAA Routing

4.1 Registration of Users Owning Smart Card Equipped Devices

Each user with a mobile device must be registered with his home network AAA server before he roams into a remote network. The mobile device is assumed to contain a smart card to securely store some secret values and perform cryptographic operations. The equipment of smart cards within mobile devices is, we believe, becoming a necessity in today's wireless technologies including GSM [7], 3G [8,9], and WiMAX [10].

From now on, based on the context, U denotes either a smart card or the user of the device. S denotes the home network AAA server with which U comes for registration. Registration is done securely via a secure channel (denoted as $\Rightarrow$) since it requires the user with his device to access to the server in person or over the provider's private network. $h()$ denotes a secure hash function with a sufficient length of hash output such as SHA-256. The registration protocol below is inspired from the work in [11], which suggested to save a hashed password for each user at the server's database.

A user can freely choose his ID and password. A mobile device prompts the user for his ID and password. The correct ID and password activates the device's smart card.

$U \Rightarrow S: id, h(a \oplus pw)$

a : a random secret chosen by the smart card U

id, pw : U's (real) ID and password

$S \Rightarrow U: m, \hat{id}$

$m = h(id \oplus x_{id} \oplus x) \oplus h(a \oplus pw)$

x_{id} : a random secret uniquely given to U by S

x: S's master secret value stored safely at S

$\hat{id}$: U's pseudo ID to be used next time

Fig. 3 User Registration via a Secure Channel

S does not store users' passwords in the clear, which prevents an attacker from directly obtaining passwords even from the server's verification table. The values m and $\hat{id}$ are stored in the smart card that also contains the value a, implementations of the RC4 algorithm and the hash function $h()$. For each of its users, S stores his ID, $\hat{id}$, x_{id} and other information to be discussed later. S securely saves the master secret x as well.

4.2 Handshake Protocol with an Access Network

We propose a handshake protocol to allow a mobile user to discover and select an access network through which to transmit authentication packet. We assume a mobile user is contacting a WLAN access network via an AP run by the network. Although it is described using a WLAN environment, the protocol is generic in nature and can be easily adapted to other wireless technologies. The protocol is inspired from the work in [2], where an efficient scheme, called RIC-VAP (Roaming Information Code-Virtual AP), was proposed to deliver available network information to the nearby mobile devices. We assume that an AP's SSID or a new IE in a beacon broadcast can encode the roaming groups to which the access network running the AP belongs. The encoded roaming group information is to fit into even a single SSID field. A mobile device within the beacon signal range then retrieves information from the signal and sees if there is a roaming group that the device's user can use. In case a device has found no roaming group to use, it repeats the process of listening to other AP's beacon signals. To choose an access network to use, the mobile user needs to store information in his device about the roaming groups to which his home network belongs and the credentials for each of these roaming groups.

$AP \rightarrow U$: $(A;B;C)$ // groups A, B, C (example)
// roaming groups in the beacon signal
$U \rightarrow AP$: $seq\_no, G, hn, preference$
// Probe Request
seq_no : a number given to this request by U
hn: U's home network
G: the roaming group selected by U that contains hn as a member. G is either $A, B,$ or C
$preference$: U's preferred metric on the route
X_k (AP) $\rightarrow U$: $seq\_no, cost, h(\alpha)$
// Probe Response
X_k : the access network belonging to G and reachable via the AP
seq_no : the sequence number copied from U's request
$cost$: the cost of the path to the destination hn over the roaming group G according to the $preference$
$h(\alpha)$: a hash value given to U for being attached to U's request via XOR.

Fig. 4 Handshake Protocol with a WLAN Access Network

The proposed handshake protocol has the desirable characteristics as follows.

(1) AP can deliver information on different access networks by a compact RIC [2]-like code via a single beacon signal, which saves bandwidth.
(2) Access networks belonging to different roaming groups may adopt different security policies and authentication schemes
(3) Mobile device needs not initiate the query process to AP for the information on access network and roaming group selection.

In the last step, X_k gives $h(\alpha)$ to U, where α is a nonce set by X_k, and $h(\alpha)$ be contributed to U's authentication request. X_k, on the receipt of U's packet toward S, would attach α via XOR to the U's request. This feature ensures X_k's participation in delivering the packet to S, since only X_k is supposed to know the value α.

4.3 Mutual Authentication Protocol for Roaming Users with Mobile Devices

We propose a mutual authentication protocol between a roaming user and his home network AAA server.

In the protocol, U's pseudo ID $\hat{id}$ to identity U at the server S changes each time, which makes it infeasible for an attacker to link any two authentication trials belonging to the same user. In addition, $\langle id, pw \rangle$, the real ID and password, never leaves the user's device in the clear. Also, note that the access network needs to attach α, a nonce for this request of U, to the Y in U's request via XOR before it forwards the request packet.

The size of U's request is about 160 octets assuming each part of the request but the *retryflag* is 256 bits long. S's response packet is 64 octets long assuming T_s and V are of size 256 bits, respectively. A timestamp is repeated as many times as necessary to fill out the 256 bits width in

case a shorter, say 16 octets, timestamp is used. We claim that the authentication packets are small enough to be fit into a single frame in any existing wireless network.

U's correct input $\langle id, pw \rangle$ activates the device's smart card

$U \rightarrow S$: $retryflag, i\hat{d}, T_u, Z, W, Y$

$retryflag$: 0 if it is the first request using $i\hat{d}$;
1 otherwise

$i\hat{d}$: U's current pseudo ID

T_u : U's current timestamp

$Z = h(m \oplus h(a \oplus pw) \oplus T_u) \oplus i\tilde{d}$

$W = h(i\tilde{d} \oplus T_u) \oplus h(\alpha)$

$Y = h(i\hat{d} \oplus h(i\tilde{d}))$

$i\tilde{d}$: U's next pseudo ID (a 256-bit random key generated by RC4) for the next and new authentication request

$h(\alpha)$: ensuring S that authentication traffic has passed through the access network contacted by U, where α is a nonce generated by the access network

$S \rightarrow U$: T_s, V

T_s : S's current timestamp

$V = h(h(id \oplus x_{id} \oplus x) \oplus i\tilde{d} \oplus h(T_s))$

Fig. 5 Mutual Authentication Protocol

The verification requires to consult S's verification table as below.

Table 3: *S*'s Verification Table

ID	pseudo ID(used)	pseudo ID(next)	secret value
id	id'	id''	x_{id}

S verifies an incoming request as follows:

(1) if T_u is not recent enough, discard the packet and stop; else continue.

(2) look up pseudo ID(next) fields (if $retryflag$ is 0), or both pseudo ID(used) and pseudo ID(next) fields (if $retryflag$ is 1) using the value $i\hat{d}$; if no matching entry found, discard the packet and stop.

(3) let id and x_{id} be the ID and the secret value, respectively, of the matching entry.

(4) compute $D = h(h(id \oplus x_{id} \oplus x) \oplus T_u)$, where T_u is the received value.

(5) compute $E = Z \oplus D$, where Z is the received value.

(6) compute $F = h(E \oplus T_u)$, where T_u is the received value.

(7) compute $G = W \oplus F$, where W is the received value.

(8) compute $H = h(i\hat{d} \oplus h(E))$, where $i\hat{d}$ is the received value.

(9) compute $I = Y' \oplus H$, where Y' is the received value of Y in U's request. Note that $Y' = Y \oplus \alpha$ since the access network relaying U's request XORs α with Y before it forwards the request.

(10) compute $h(I)$ to see if it equals to G : *equal* - (success) store $i\hat{d}$ in the pseudo ID(used) field and E in the pseudo ID(next) field, respectively; *not equal* - (failure) discard the packet and stop.

U verifies S's response as follows:

(1) if T_s is not recent enough, discard the packet and stop; else continue.

(2) compute $J = m \oplus h(a \oplus pw)$, where a is the value stored at the smart card and pw is the user input value

(3) compute $K = i\tilde{d} \oplus h(T_s)$, where $i\tilde{d}$ is the value stored at the smart card and T_s is the received value.

(4) compute and see if $h(J \oplus K)$ equals to the received V : *equal* - (success) remove the stored $i\hat{d}$ and set $i\tilde{d}$ as the next ID to be used; *not equal* - (failure) stop. Re-try with the same access network or with a different access network using $i\hat{d}$ again.

S's response packet follows the path taken by the request packet to which it responds in reverse order. U will wait for his previous authentication request to be acknowledged by S within a reasonable time. If necessary, U will resend the pseudo ID of the previous request since U has no idea if his previous request including his new pseudo ID has been accepted or not at S. S has to keep the user's previous ID used as well as his new ID for the next time and to decide which ID to use for verification. To inform S of its status, U's request has a $retryflag$ bit saying if

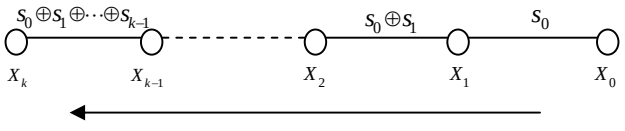
it is a new request (0) or a re-request with the same ID (1).

4.4 Route Deviation Detection

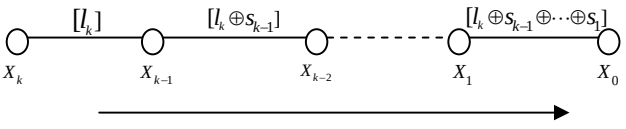
For a path $X_k, X_{k-1}, \cdots, S(=X_0)$, where X_k is the access network contacted and selected by U, each node X_i $(1 \le i \le k)$ but the destination S contains a unique label $l_i = s_0 \oplus s_1 \oplus \cdots \oplus s_{i-1}$ for the path. Note that each node must maintain such values for every path it stores. X_i $(0 \le i < k)$ also stores a random secret s_i for the edge $\langle X_{i+1}, X_i \rangle$. Figure 6 shows the forward direction of a label construction discussed above and the backward packet delivery that effectively undoes the label construction process and recovers an initial secret value at the initiating node X_0. It helps to understand how the route deviation detection works.

The access network X_k chooses a random value α for a particular request of U and XORs it with the Y in U's request. In addition, X_k XORs its label, l_k, of the path to S with the Y in U's request to get the value $Y^* = Y \oplus \alpha \oplus l_k$. Then, each X_i $(1 \le i < k)$ receives Y^* from X_{i+1} and delivers $Y^* \oplus s_i$ to X_{i-1}. Then the value finally arrived at S $(=X_0)$ will be $\bar{Y} \equiv Y \oplus \alpha \oplus l_k \oplus s_{k-1} \oplus \cdots \oplus s_1$. Now replace the step (9) of the verification process at S with the step (9'), where

(9') is: *compute* $I = \bar{Y} \oplus s_0 \oplus H$ *, where* $\bar{Y}$ *is the received value and* s_0 *is the secret value of* S $(=X_0)$ *for the edge* $\langle X_1, X_0 \rangle$.



(a) label construction: X_k stores $l_k = s_0 \oplus s_1 \oplus \cdots \oplus s_{k-1}$



(b) packet delivery: X_0 receives $l_k \oplus s_{k-1} \oplus \cdots \oplus s_1 = s_0$

Fig. 6 Forward Label Construction and Backward Packet Delivery

In the delivery, $Y^* \oplus s_i$ is encrypted and delivered as $E_{M_{i,i-1}}\{Y^* \oplus s_i\}$, where $M_{i,i-1}$ is the secret key securely established between X_i and X_{i-1}. With this feature add-on to the authentication protocol, the server is able to detect any deviation or missing node on the route taken by U's authentication traffic.

5. Security Analysis

This section analyzes the security of the solution against various attacks attempting to compromise the protocols. Throughout the discussion, we argue that a brute force attack to the hash function $h()$ is computationally infeasible due to its output size of 256 bits.

5.1 Attacks for Obtaining Secret Information

(1) by reading S's verification table: Even if S's verification table is disclosed, it is infeasible to obtain any secret information directly usable to impersonate users unless the master secret x of S is also compromised.

(2) by eavesdropping authentication request and/or response: Suppose an attacker intercepts U's request packet. In the packet, $retryflag, T_u$ and $i\hat{d}$ are in the clear, whereas Z, W and Y are protecting their components as follows. Z: It involves the next pseudo ID of U. Even if the attacker is able to obtain the ID somehow, it is infeasible to get U's m, a, or pw from $h(m \oplus h(a \oplus pw) \oplus T_u)$ due to $h()$'s one-way property. W: With the knowledge of $i\tilde{d}$, which is never easy to get, the attacker is able to compute $h(i\tilde{d} \oplus T_u)$ and $h(\alpha)$ in turn. However, both of these are useless to the attacker. A timestamp and a one-time pseudo ID folded into $h(i\tilde{d} \oplus T_u)$ will change next time. And the probability of reusing $h(\alpha)$ is negligible since α is a nonce with a sufficient size. Y: Even if the attacker is able to isolate and obtain α from Y, it is very unlikely for the same α to be used again. Hence it does no good to the attacker. Moreover, Y in the request packet is encrypted by each mediating network to prevent the attacker from deducing the secret value s_i by comparing inward and outward packets of a network. The attacker may want to use S's response to retrieve some secret information. But, it is infeasible to extract the secret part $id \oplus x_{id} \oplus x$ from V

and/or T_s due to $h()$'s one-way property.

(3) by extracting information from user's smart card: Assume a registered user U somehow extracts m and a from his smart card, and then retrieves $h(id \oplus x_{id} \oplus x)$ by XORing m with $h(a \oplus pw)$. But it is computationally infeasible for U to get $id \oplus x_{id} \oplus x$ from $h(id \oplus x_{id} \oplus x)$. U is able to obtain the master secret x only when (i) U succeeds in (i) computing $id \oplus x_{id} \oplus x$ from $h(id \oplus x_{id} \oplus x)$, and (ii) obtaining his own secret x_{id} from S's verification table. We disregard such a possibility.

5.2 Impersonating the Server

To impersonate S to U, an attacker needs to derive V and T_s, where the attacker can easily select a timestamp T_s and compute $h(T_s)$. For deriving V that can pass the verification process at U, the attacker has to know U's next pseudo ID to be used since V must be equal to $h(h(id \oplus x_{id} \oplus x) \oplus i\tilde{d} \oplus h(T_s))$. Among the terms comprising V, $i\tilde{d}$ will not be sent in the clear until U verifies the response of the attacker who claims to be S. Moreover, it is unlikely that the attacker is able to retrieve or compute $h(id \oplus x_{id} \oplus x)$. Therefore it is infeasible for the attacker to fabricate a V looking valid to U.

5.3 Modifying Packets Illegally

(1) modifying $retryflag, T_u$ and/or $i\hat{d}$: T_u and $i\hat{d}$ are used to compute (Z,W) and Y, respectively. To deceive S for accepting the modified packet as a valid one, an attacker has to find an input value that produces the same hash output as (Z,W) or Y, which is computationally infeasible. So, modifying T_u and/or $i\hat{d}$ will be detected by the verification process at S. Toggling the value of $retryflag$ makes S verify the user with a wrong ID, which will result in a verification failure.

(2) modifying Z,W and/or Y: It will be detected at S as well, because any modification on the input of $h()$ almost certainly makes a different hash output. For the value α selected by the access network, however, it is possible for the access network to change the value arbitrarily, say α', without being detected. It is because the access network itself has generated and sent the computed value $h(\alpha)$ to U who used the value for computing W. However, it will not do the access network any good.

(3) modifying V and/or T_s: Tampering with T_s will be detected by U's verification process since it affects $h(T_s)$ in V. Any attempt to modify V will also be detected because it is unlikely for the modified value to make a valid looking V.

5.4 Reflection Attack

The attack is to fabricate packet data for the same or another session that will be able to fool the receiver. Relevant data are collected from the passing authentication packets. For example, if an attacker is able to derive V immediately from U's authentication request $(retryflag, i\hat{d}, T_u, Z, W, Y)$, the attacker could deceive U by pretending to be S. To prevent such an attack, the packet components are devised so as to make it infeasible to be derivable from other components.

(1) Deriving V (response) from Z,W and/or Y (request):

- deriving V from Z: To make it look like a valid V, an attacker needs to obtain and fold $i\tilde{d}$ into $h()$. However, obtaining $i\tilde{d}$, the next pseudo ID of U, is very hard since it was never sent before in the clear. Moreover, for the given $i\tilde{d}$, the chance of T_u being equal to $i\tilde{d} \oplus h(T_s)$ is extremely low, where T_s must be a recent and valid timestamp.
- deriving V from W: An attacker is not able to obtain the value α (and $h(\alpha)$) unless the access network itself is the attacker. Even with the value $h(\alpha)$, the attacker further needs to get $h(id \oplus x_{id} \oplus x)$ or make T_u equal to $h(id \oplus x_{id} \oplus x) \oplus h(T_s)$ for a recent and valid timestamp T_s. Either possibility is negligible.
- deriving V from Y: Even if an attacker comes to know α, the attacker further needs to have $h(id \oplus x_{id} \oplus x)$ and $i\tilde{d}$ to generate a valid looking V, which are simply not derivable from Y.

(2) deriving $[Z,W,Y]$ (request) from V (response): Z,W and Y are all containing $i\tilde{d}$, the next pseudo ID of

U, which is very hard to get. In addition, to derive Z from V, an attacker needs to use a timestamp T_u' that must be equal to $i\tilde{d} \oplus h(T_s)$, where the chance of it being accepted as a valid timestamp is negligible. Since Z, W and Y are always sent together, it would be of no use to produce just one or two of them.

5.5 Making Authentication Fail

The purpose of this attack is to make certain or all authentications fail. An attacker can easily do that by arbitrarily modifying the passing traffic. We have no effective prevention against this denial of service type attack but simply avoiding the networks on the route through which several authentication trials failed.

6. Related Work

A closely related issue to AAA routing is identity selection, where user selects which identity and credentials to use for authentication in a given point of attachment. In the process, some information should be given to the user who is supposed to make a final decision. In WLAN access networks, the ways of exchanging such information include SSID in beacon, EAP Identity/Request [12,13], and RIC [2]. It also varies how to structure information to be delivered to the user. The RIC [2] work proposed a couple of encoding schemes that accomplish a very compact way of representing numerous roaming groups and related information. Other structuring techniques suffer from the restricted payload space and thus are capable of holding information about a limited number of networks only. In cellular networks, a list of feasible visited networks to select from are pre-prioritized and stored in the SIM [7] card of a mobile device.

To relieve excessive burden of data collection, analysis, and/or maintenance imposed on user device, [3] proposed a system architecture that uses various network entities to assist the user in choosing an appropriate network. In [4], the same authors presented a decision making process that can effectively rank candidate networks for user in the network architecture proposed in [3]. They also proposed an approach [5] for network selection when imprecise information was given for the decision. Our work has a different problem domain of AAA routing, which was not dealt with in [3,4,5].

Once an identity has been selected, it needs to route the authentication traffic originating from the selected identity back to the home network. A widely used source routing approach is based on the domain in a given NAI (Network Access Identifier)[14]. A decorated NAI is a NAI, that is, a user identity with his domain, with additional information on the route to the home network server. It specifies the mediating networks to pass as a prefix to the NAI followed by /. For example, mediating-net-1.com/mediating-net-2.com/username@homerealm.com specifies a series of mediating networks to pass including the final destination home network. Packets with NAI-based routing hints are processed by network servers and proxies running RADIUS [15] or Diameter [16,17]. However, these packets are subject to spoofing and modification. Therefore the home network server has no way to verify that the path of mediating networks used was the same one that the user has requested. The solution in this paper is unique in the sense that it ensures the authentication of the route used as well as the user privacy by using a one-time identifier.

7. Conclusion

Roaming over various heterogeneous networks such as 3G, WLAN and WiMAX requests a mobile user to be authenticated and authorized before the user receives desired service. Based on a graph model of roaming group, we proposed a secure and lightweight solution that helps a roaming user to select which route and identity to use and to exchange authentication traffic with his home network AAA server. Our network-based solution provides and suggests an appropriate roaming path for each preferred metric. To support that, we presented an algorithm for routing and label computation in a graph representing a roaming group of provider networks. Unlike the user-based approach, in which the user himself collects and analyzes data, the proposed solution burdens the user with no extra work for selecting appropriate networks. Moreover the solution is using very fast operations only such as XOR, SHA-256, and RC4 algorithm. The exchanged packet size is less than 161 octets, which eliminates the possibility of packet fragmentation in wireless links. Certainly these characteristics are beneficial to battery-powered mobile devices. It guarantees user privacy that makes it infeasible to link any of two verification requests. The solution also allows the home network server to detect any deviation or missing of nodes on the route used. By carefully composing a set of related protocols, the solution successfully resolved all major issues discussed above, which was never done before.

The proposed solution is generic and flexible since it

requires no special authentication credential but password, transmission mechanism, or payload. It does not require any change on the IP routing infrastructure, either. The solution could be deployed incrementally over some AAA servers to interoperate with existing AAA infrastructure by tunneling an island of provider networks not knowing of the solution. We plan to investigate if the authentication method might be applicable to handoff situation where quick re-authentication or delegation of authority seems to be inevitable.

Acknowledgments

This work was supported by the IT R&D program of MKE/IITA. [2006-S-041-03, Development of a common security core module for supporting secure and trusted service in the next generation mobile terminals]

References

[1] J. Arkko, B. Aboba, J. Korhonen, and F. Bari, Network Discovery and Selection Problem, RFC 5113, January 2008.

[2] Y.-W. Lee and S. C. Miller, "Network Selection and Discovery of Service Information in Public WLAN Hotspots," 2nd ACM International Workshop on Wireless Mobile Applications and Services on WLAN Hotspots(WMASH 2004), October 2004, pp. 81-92.

[3] F. Bari and V. C. M. Leung, "Service Delivery over Heterogeneous Wireless Networks: Network Selection Aspects," Proc. ACM Int'l Wireless Communications and Mobile Computing Conference, July 2005, pp. 251-256.

[4] F. Bari and V. C. M. Leung, "Automated Network Selection in a Heterogeneous Wireless Network Environment," IEEE Network, January/February 2007, pp. 34-40.

[5] F. Bari and V. C. M. Leung, "Network Selection with Imprecise Information in Heterogeneous All-IP Wireless Systems," Proc. ACM Int'l Conf. on Wireless Internet, October 2007.

[6] O. Ormond, P. Perry, and J. Murphy, "Network Selection Decision in Wireless Heterogeneous Networks," Proc. IEEE Int'l Symposium on Personal, Indoor and Mobile Radio Communications, 2005, pp. 2680-2684.

[7] H. Haverinen and J. Salowey, Extensible Authentication Protocol Method for Global System for Mobile Communications (GSM) Subscriber Identity Modules (EAP-SIM), RFC 4186, January 2006.

[8] 3GPP, "3GPP System to Wireless Local Area Network (WLAN) interworking; WLAN User Equipment (WLAN UE) to network protocols; Stage 3 (Release 7)," 3GPP TS 24.234 V7.2.0, June 2006.

[9] 3GPP, "3G Security; Wireless Local Area Network (WLAN) interworking security (Release 7)," 3GPP TS 33.234 V7.1.0, June 2006.

[10] WiMAX Forum, "Mobile WiMAX-Part I: A Technical Overview and Performance Evaluation," http://www.wimaxforum.org/technology/downloads/Mobile_WiMAX_Part1_Overview_and_Performance.pdf, August 2006.

[11] H. Chien, J. Jan, and Y. Tseng, "An Efficient and Practical Solution to Remote Authentication: Smart Card," Computers and Security, Vol. 21, No. 4, pp. 372-375, 2002.

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

[13] F. Adrangi, V. Lortz, F. Bari, and P. Eronen, Identity Selection Hints for the Extensible Authentication Protocol (EAP), RFC 4284, January 2006.

[14] B. Aboba, M. Beadles, J. Arkko, and P. Eronen, The Network Access Identifier, RFC 4282, December 2005.

[15] B. Aboba and P. Calhoun, RADIUS(Remote Authentication Dial In User Service) Support For Extensible Authentication Protocol (EAP), RFC 3579, September 2003.

[16] P. Calhoun, J. Loughney, E. Guttman, G. Zorn, and J. Arkko, Diameter Base Protocol, RFC 3588, September 2003.

[17] P. Eronen, T. Hiller, and G. Zorn, Diameter Extensible Authentication Protocol (EAP) Application, RFC 4072, August 2005.

Jun-Cheol Park received his Ph.D. in Computer Science from the University of Maryland, College Park, USA, in 1998. He is an associate professor with the Department of Computer Engineering, Hongik University, Korea. His research interests include network security, overlay networks, and wireless mesh networks.

Sung Ik Jun received his MS and BS in Computer Science from Chung-Ang University, Korea, in 1985 and 1987, respectively. He joined ETRI (Electronics and Telecommunications Research Institute) in 1987, and since then he has been conducting R&D on various aspects of embedded real-time systems. Currently, he is the head of the Wireless Security Research Team. His research interests are wireless security, smart cards, wireless sensor networks, and real-time systems.