Towards Pervasive Computing Security

L. A. Mohammed

Abstract - **The change in physical structures of computing facilities into small and portable devices or even wearable computers has enhanced ubiquitous information processing. The basic paradigm of such pervasive computing is the combination of strongly decentralized and distributed computing with the help of diversified devices allowing for spontaneous connectivity via the Internet. In general, pervasive computing strives to simplify day-to-day life by providing mobile users with the means to carry out personal and business tasks via mobile and portable devices. This paper examines the security challenges that are barriers to mainstream pervasive computing and proposes some countermeasures. In particular, the paper focuses more on challenges associated with ARP poisoning where IPv4 network is used.**

Index Terms **- Pervasive Computing, ARP poisoning, Mac Spoofing, Wireless and Mobile Security, Hacking, Flooding and Spoofing Attacks**.

I. INTRODUCTION

Today, mobile phones, PDA and similar devices are arguably the dominant computer form factor consumers' purchase. These devices have become powerful and sophisticated, many are even more powerful than desktop computers of the late 1990s [1] They are capable of receiving TV and cable network services, radio station services and other audio-visual services in addition to communication services. Technologies like Bluetooth and Wi-Fi make it possible to embed networking capabilities into any small devices without hassle [2] In effect, these technologies help make networking much more general and achievable even on elementary devices, like toasters and paperclips. In such computing environments, these services will increase both the complexity of information infrastructures and the networks which support them. However, Information stored, processed, and transmitted by the various devices is one of the most critical resources. Threats exploiting vulnerabilities of new kinds of user interfaces, displays, operating systems, networks, and wireless communications will cause new risks of losing confidentiality, integrity, and availability.

In this paper we organize and present various security challenges associated with the pervasive computing and also proposed some countermeasures. In particular, we look at both ARP poisoning and Mac spoofing attacks.

Manuscript received March 21, 2008.This work was supported in part by King Fahd University of Petroleum and Minerals, Saudi Arabia.

L. A. Mohammed is with Computer Science and Engineering Technology Unit, ,King Fahd University of Petroleum and Minerals, HBCC Campus, King Faisal Street, Hafr Al Batin 31991, Saudi Arabia, (e-mails: gumel@hbcc.edu.sa)

Other issues such as virus and malware were also briefly discussed. The rest of the paper is organized as follows: Section II describes some generic security challenges in pervasive computing environment. Description of ARP and DHCP protocols were given in section III. Section IV briefly explains the ARP poisoning attack. Section V explains the proposed countermeasure protocol. Finally, section VI concludes the paper.

II. SECURITY CHALLENGES IN PERVASIVE COMPUTING ENVIRONMENT

Pervasive computing environment or PCE share most of the security issues of traditional networked applications. However, the pervasive computing environment adds some unique issues to the already complex security arena. Physical security is important as the devices can be easily misplaced or stolen. Information that is usually confined behind a corporate firewall is now winging its way through the air, possibly spending some time on hosted servers or wireless gateways.

The techniques of hacking mobile devices such as laptops, cell phones, PDAs etc is already spreading. In view of these, adding security to such environment presents challenges at different levels. Authenticating the identity certificate of a previously unknown user doesn't provide any access control information. Simple authentication and access control are only effective if the system knows in advance which users are going to access a particular subject or stored information and what their access rights are. Table 1 below describes some hacking tools for mobile and wireless devices.

In addition to hacking, malicious codes such as viruses, trojans, worms, and spyware can load themselves onto wireless devices and run without user knowledge or action. The successful installation and operation of a simple malware program can effectively use all available memory and halt device performance. A more dangerous malicious program can transmit itself across the wireless network, bypassing some of the corporate network security systems, and potentially damaging other components of the corporate network.

Examples of mobile devices malicious codes that can be transmitted include the following [3]:

1. *Phage* (a virus)- when executed it will overwrites third-party Palm OS application programs which will then no longer function as designed.
2. *Vapo*r (Trojan horse) – as the name implies, this can hide itself and also other applications from the user so that they appear to have been deleted.
3. *Spammed message* (such as Compact HTML for NTT DoCoMo phones) – this message disguises in the form of paging asking the user to select an option, which will then run a particular script.

4. *Timofonica* (worm) – this worm spread from PCs to PCs via e-mail attachment, the e-mail message can be received by a mobile device via Telefomica's (Spain's Telecommunication) GSM gateway. It can erase system information and leave the machine unable to boot!
5. *Liberty Crack* – This is a Trojan horse that removes third-party programs from the target device. Initially, the Trojan crack appeared in Internet Relay Chat groups, then spread to the Web and newsgroups, from which users could download the program.

Table 1: Hacking tools for wireless devices

Tools Description
Airjack - DoS tool that sends spoofed authentication frames to an AP with inappropriate authentication algorithm and status codes.
AirSnort - A tool which recovers encryption keys. It operates by passively monitoring transmissions and computing the encryption key
Bloover II - It is a J2ME-based auditing tool. It is intended to serve as an auditing tool to check whether a mobile phone is vulnerable.
BlueBugger - This exploits the BlueBug vulnerability. One can gain an unauthorized access to the phone-book, calls lists and other private information.
Bluediving - It is a Bluetooth penetration testing suite. It implements attacks like Bluebug, BlueSnarf, BlueSnarf++, BlueSmack
Bluesnarfer - This tool can be used to download phone-book of any mobile device vulnerable to Bluesnarfing.
BlueSniff - A GUI-based utility for finding discoverable and hidden Bluetooth-enabled devices.
BlueTest - A Perl script designed to do data extraction from vulnerable Bluetooth-enabled devices.
BTAudit – This is a set of programs and scripts for auditing Bluetooth-enabled devices.
BTBrower – An application that can browse and explore the technical specification of surrounding Bluetooth-enabled devices.
BTCrack 1.1- This is a Pass Phrase (PIN) cracking tool. The tool would let an attacker that grabs the PIN to decrypt the victim's traffic and gain full access to each of the connected Bluetooth devices.
BTCrawler - This is a scanner for Windows Mobile based devices. It can be used to implements BlueJacking and BlueSnarfing attacks.
Ettercap - Suite for *Man-in-the-Middle* attacks. It features sniffing of live connections and content filtering on the fly.
Hidattack - It let attackers hijack a Bluetooth keyboard, and the other similar devices. It basically attacks the Bluetooth human interface driver (HID) protocol.
IRPAS - A Routing Protocol Attack Suite designed to attack common routing protocols including CDP, DHCP, IGRP and HSRP.
MeetingPoint - A tool use to search for bluetooth devices. It can be combine it with any bluejacking tools to perform more serious attack.
Ministumbler - A tool for finding open wireless access points fro wardrivin. This is a WinCE version of Netstumber for PDAs
T-BEAR - This is a security-auditing platform for Bluetooth-enabled devices. The platform consists of Bluetooth discovery tools, sniffing tools and various cracking tools.
WiFiDEnum - Tool use to scan Windows hosts over the network, extracts registry information to identify wireless drivers that are installed and the associated version information.

Recently, several antivirus companies have begun to release products for handhelds. Examples include F-Secure Antivirus for WAP Gateways product which checks for malicious code at the gateway between the IP network and the WAP mobile network, then keeps it from reaching a handheld device. Similarly, McAfee's VirusScan for Handheld Devices product prevents the transmission of known PDA viruses and catches any that may already reside on the PDA. The product, which supports a number of mobile platforms, scans for known virus signatures

III. THE ARP AND DHCP PROTOCOLS

A. *The Address Resolution Protocol (ARP)*

Address Resolution Protocol (ARP) which is defined in RFC 826 [4] is used to map the IP addresses onto the data link layer MAC address. There are four main types of messages in the ARP protocol. These are identified by four values in the "Opcode" field of an ARP message, these are [5]:

a) ARP Request – When a host sends an ARP request it fills in its MAC address, IP address, type of ARP message and the target IP address. The ARP request is broadcast to all the hosts in the same LAN as the sending host. The target Mac address is left blank for the host with the target IP address to fill in.
b) ARP Reply – When a host receives an ARP request containing its own IP address as the target IP address, it fills in the target Mac address field with its MAC address. The host creates an ARP reply with the values of the sender and target fields in the ARP request reversed and the Operation field set to the opcode of the ARP reply. This packet is then sent only to the requesting machine.
c) RARP Request – Reverse Address Resolution Protocol (RARP) is the reverse of ARP. A RARP request is sent when a machine wants to get the IP address that corresponds to its MAC address. RARP requests are broadcast in the LAN.
d) RARP Reply – RARP Reply is sent by RARP servers. If the MAC address in the RARP request belongs to one of the clients served by the RARP server, a reply is sent with its corresponding IP address. RARP was later replaced by BOOTP (Bootstrap Protocol) and DHCP protocol.

Some optimizations are possible with ARP. Suppose two computers A and B are on the same LAN. Once computer A gets the ARP reply from computer B, it stores that IP-to-MAC address mapping of B in a local cache. So if in a short period of time, if A wants to communicate with B, it refers to the local ARP cache, eliminating a second broadcast. Usually, A would include its IP-to-MAC address mapping in the ARP packet, thus informing B of its mapping. In fact all machines on same LAN can enter this mapping information on A into their ARP cache. Another optimization is to have every computer broadcast its mapping when it boots, in the form of an ARP looking for its won address. To allow for changes in mapping, especially when network card breaks down, and is replaced with a new one, entries in ARP cache should time out after few minutes. If computer A is transmitting to computer C on different LAN, then it may have to use *proxy ARP*. Using normal ARP would fail as routers don't forward Ethernet level broadcasts. So A will direct all its traffic for C to a Router R with an

ARP cache entry of <IP_D, MAC_R>. R would thus handle all remote traffic. Alternatively, R can be configured to respond to ARP request for different LAN or other local networks. The ARP request message ("who is A.A.A.A tell B.B.B.B", where A.A.A.A and B.B.B.B are IP addresses) is sent as a broadcast message. It reaches all systems in the LAN as it is a broadcast. This would make sure that the target of the query will also receive a copy of the request message. Only the target system responds and the others discard the packet. The target system creates an ARP response ("A.A.A.A is hh:hh:hh:hh:hh:hh", where hh:hh:hh:hh:hh:hh is the MAC source address of the computer with the IP address of A.A.A.A). The response packet is then unicast to the address of the computer which sent the ARP query (in this case B.B.B.B). Since the original request also included the hardware MAC address of the requesting computer, it doesn't require another ARP message to find this out [6].

B. *The Dynamic Host Configuration Protocol (DHCP)*

DHCP stands for 'Dynamic Host Configuration Protocol' and is a way by which networked computers get their TCP/IP networking settings from a central server. Dynamic Host Control Protocol (DHCP) is defined in RFC 2131 [3] and 2132 [4]. It is an extension of BOOTP, the previous IP allocation specification. It allows manual and dynamic IP address assignment to computers that requests for that. DHCP server is not reachable by broadcasting from a different network. Hence a *DHCP relay agent* is needed to forward the DHCP DISCOVER broadcast packet from a newly booted machine. It is send as a unicast transmission to the DHCP server (which may be on another network) by the relay agent. The relay agent usually keeps the IP address of the DHCP server. Thus the relay agent is for relaying packets between servers and clients. This makes the DHCP server handle the sub-net that has no server available and thus there is no need to setup a server per sub-net. To keep track of the duration of IP address assignment, a DHCP server uses the concept of *leasing*. As mentioned before, the DHCP server assigns IP addresses automatically from a pool of IP addresses. If a compute leaves the network 'abruptly' and does not return the IP address that it was using, that IP address is lost for any further assignment. As a precaution to that, assignment of IP address is only for a fixed duration of time, called leasing. Just before the expiry of the lease, a computer should request the DHCP server for renewal. Otherwise, that IP address cannot be used further [7].

IV. SECURITY ATTACK -ARP POISONING PROCESS

ARP poisoning is an effective form of attack by a hacker, where by he can masquerade a network and fool the sending host. This happens because the ARP broadcast reaches him too once connected to the wired LAN or listen to wireless LAN. Later, he can reply the ARP request with a forged ARP response putting his computer's MAC address in that. The sending host is thus fooled into sending all the packets to the hacker's computer which can be forwarded to the receiving host, if needed. The attacker can also poison the receiving host and get a reverse path going. The attacker thus realizes a two way man in the middle, where he can forward the received packets to the correct destination after inspecting and possibly modifying them. The two end points of the connection will not notice the extra hop added by the attacker if the packet TTL is not decremented [8],[9]. One successful and effective attack on wireless LAN or normal LAN is ARP poisoning. That's why we propose a new ARP protocol that could mitigate such attacks.

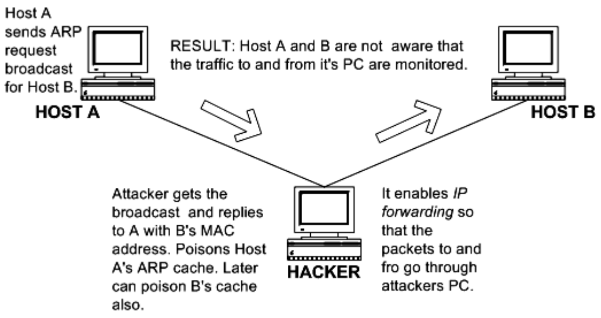
Host A sends ARP request broadcast for Host B.
HOST A
RESULT: Host A and B are not aware that the traffic to and from it's PC are monitored.
HOST B
Attacker gets the broadcast and replies to A with B's MAC address. Poisons Host A's ARP cache. Later can poison B's cache also.
HACKER
It enables *IP forwarding* so that the packets to and fro go through attackers PC.

Figure 1. The ARP poisoning mechanism.

V. SECURE UNICAST ARP (S-UARP) PROPOSAL

A paper [9] on Secure ARP (S-ARP) has been published by D. Bruschi et al. which deals with ARP broadcast communication security. Here each host has a public/private key pair certified by a local trusted party on the LAN, which acts as a Certification Authority. Messages are digitally signed by the sender, thus preventing the injection of spurious and/or spoofed information. It has been implemented also in Linux [9]. The S-UARP proposal we make is unicast in nature and have different options for security implementation.

Many organizations would have implemented a DHCP server for dynamic IP address assignment to individual machines in a LAN. Hence the DHCP server can be configured to have the MAC-to-IP address mapping or vice-versa for all the computers/hosts under its domain. We propose to extend the DHCP protocol to handle Secure Unicast Address Resolution Protocol (S-UARP) packets. We denote such a server as DHCP+ server from now on. The DHCP relay agent also needs to be modified to forward the S-UARP request/response messages. When using dynamic IP addressing using DHCP, the DHCP+ server stores the mapping of IP to MAC address as it leases out the IP address to the requesting hosts. We are not dealing with static IP addressing option here. But some suitable modification to this protocol can make it suitable for static addressing too. The proposal itself has an inherent partial-security against eavesdropping compared to ARP broadcast in a wired network, since packets are unicast in nature and is not broadcasted. In a wireless network, a packet sniffer can capture these unicast packets too since the radio transmission has no defined boundaries of transmission. But we add security into our protocol proposal.

A. S-UARP Protocol

This is a centralized protocol unlike the decentralized approach in normal ARP. Consider the following notations and their meaning as shown below.

Notation:	*Meaning:*
S-UARP_req:	S-UARP Request Packet
S-UARP_res:	S-UARP Response Packet
DHCP+:	DHCP+ Server
ICP:	Integrity Check Pass (security flag)
ICF:	Integrity Check Fail (security flag)
A:	Host A
B:	Host B
IP_A:	IP address of A
MAC_A:	MAC address of A
IP_B:	IP address of B
MAC_B:	MAC address of B
S_K:	Session key
K_{SA}:	Shared secret key between host A and the server
MIC:	Message Integrity Code
H	Collision Free One-Way Hash Function
t:	Time (independent variable) with one or more independent values.
t1:	Time period (duration) when receiver waits for S-UARP_req
t2:	Time period when sender looks for a packet to be sent to the same host where ACK has to be sent.
t3:	Time period within ACK packet has to be sent. (t3 > t2)
t4:	Time period after which S-UARP cache needs refreshing.

The S-UARP protocol is described as follows in 3 steps:

1. A → DHCP+ : S-UARP_req
2. DHCP+ → A : S-UARP_res + MIC
3. A → DHCP+ : (ACK) K_{SA}

A simple example and explanation to show how this can be implemented with DES algorithm is as follows:

1. When a host *A* wants to communicate to host *B*, it sends a S-UARP request packet (unicast packet) to the DHCP+ server (which answers the S-UARP packets), instead of sending a broadcast to all. We assume that the secret hashing key (K_{SA}) is distributed between the client and the server, using private-public key mechanism or any other secure mechanism.
2. The DHCP+ server encrypts the response message using DES with cipher block chaining (CBC). It cuts the message (S-UARP_res) into predetermined-sized of *i* blocks (where *i = 1, 2, …., n*). Use the CBC residue (that is the last block output by CBC process) as a message integrity code (MIC). This MIC would act as a checksum [7]. The plaintext message plus the MIC would be transmitted to the host (receiver) or A. i.e. DHCP+ Server → Host A: Transmit S-UARP response (plain text) + MIC. The transmitted response message will be as follows:

S-UARP response (plain text)	MIC (CBC residue)

Figure 2. The S-UARP response message and MIC transmitted from DHCP+ Server.

If the response message doesn't arrive within a time period *t1*, host A will retransmit another S-UARP request packet to server. This can continue until it gets a request packet.

3. Once the UARP response is received, host A checks for validity by using its secret key. The receiver (Host A) encrypts the plaintext S-UARP_res using DES that it received with the shared secret key and do the hashing process to produce similar MIC (say, MIC*). Finally it checks the CBC residue or MIC. If MIC = MIC*, the message is a non-tampered in transit. We then call it Integrity Check Pass (ICP) state. Otherwise it is Integrity Check Fail (ICF) state and is discarded. The S-UARP response contains time t_s when it was generated by the server. Host A also checks the freshness of the response by checking $t_r - t_s = \Delta t$ (similar to t3), where t_r is the time when A receives the response from the server and Δt is the accepted time interval for transmission delay. Finally, the host A sends an encrypted acknowledgment (ACK)K_{SA} to the server. ACK contains the timestamp t_a generated by the host A to ensure that the message is fresh and is not a replay.

The entries in S-UARP cache remains valid for a time period, *t4* (say, in minutes) as in ARP protocol. Once that time period expires, a new S-UARP request need to be sent by a host to DHCP+ server to get the IP-to-MAC address mapping. This can deal with a situation of change in ethernet card for a machine.

B. Detailed Explanation

The protocol can be shown in detail as follows, with the timing details and optimization (as explained under section IV.C). When DHCP+ Server assigns a dynamic IP address to a host, the IP and MAC address of the DHCP+ server should be made known to the host.

```
Procedure S-UARP_Communication (A→B)
BEGIN:
Initialize the flag [pkt_send (from → to)] = failure;
while (pkt_send (A → DHCP+)  == failure)
{
  Initialize t;
  S-UARP_req (IP_A, MAC_A, IP_B);
  A → DHCP+  : Sends S-UARP_req;   //no broadcast
  if (t < t1)
```

```
        pkt_send (A → DHCP+) = success;
      else
        pkt_send (A → DHCP+) = failure;
  }
  while (pkt_send (A → DHCP+) == success || t > t3))
  {
    Initialize t;
    S-UARP_res (IP_A, MAC_A, IP_B, MAC_B, $t_s$)
    DHCP+ → A  : Sends UARP_res + MIC;
    if (pkt_send (DHCP+ → A) == success && t < t2
       && ICP )
    {
      Host A → DHCP+: Piggyback (ACK)$K_{SA}$;
      if (pkt_send (A → DHCP+) == success)
        S-UARP Cache updated;
      else
        Go to start of enclosed while loop; flag = success;
      A → B : A communicates to B directly;
    }
    else if (pkt_send (DHCP+ → A) == success &&
        t2 < t < t3 && ICP)
    {
      Host A → DHCP+: Sends (ACK)$K_{SA}$ packet;
        if (pkt_send (A → DHCP+) == success)
        S-UARP Cache updated;
      else
        Go to start of enclosed while loop; flag = success;      A → B : A communicates to B directly;
    }
    else if (pkt_send (DHCP+ → A) == failure || t > t3)
    {
        Go to start of enclosed while loop;
    }
  }
  if (t > t4 || ICF )
    S-UARP_Communication (A→B);
  END: //end of procedure
```

C. *Possible Optimization*

An optimization possible is that the ACK can be piggybacked on another packet to the DHCP+ server, if packet transmission from host A to server happens within time *t2*. This can eliminate the separate ACK packet sent and save ACK congestion in the network. If there is no scope for piggybacking, and the acknowledgement is not received within a reasonable time period *t3* (where t3 > t2), the server sends the S-UARP response packet again. If the S-UARP response packet is received by the host and the ACK packet is lost on transit, the duplicate response packets send by the server (after timeout *t3*) would be rejected.

D. *Flow Chart for S-UARP*

The flow chart for the S-UARP protocol can be shown as follows. It depicts the scenario when Host A wants to communicate to Host B (or a general Host X) and how the protocol works with respect to different time durations.

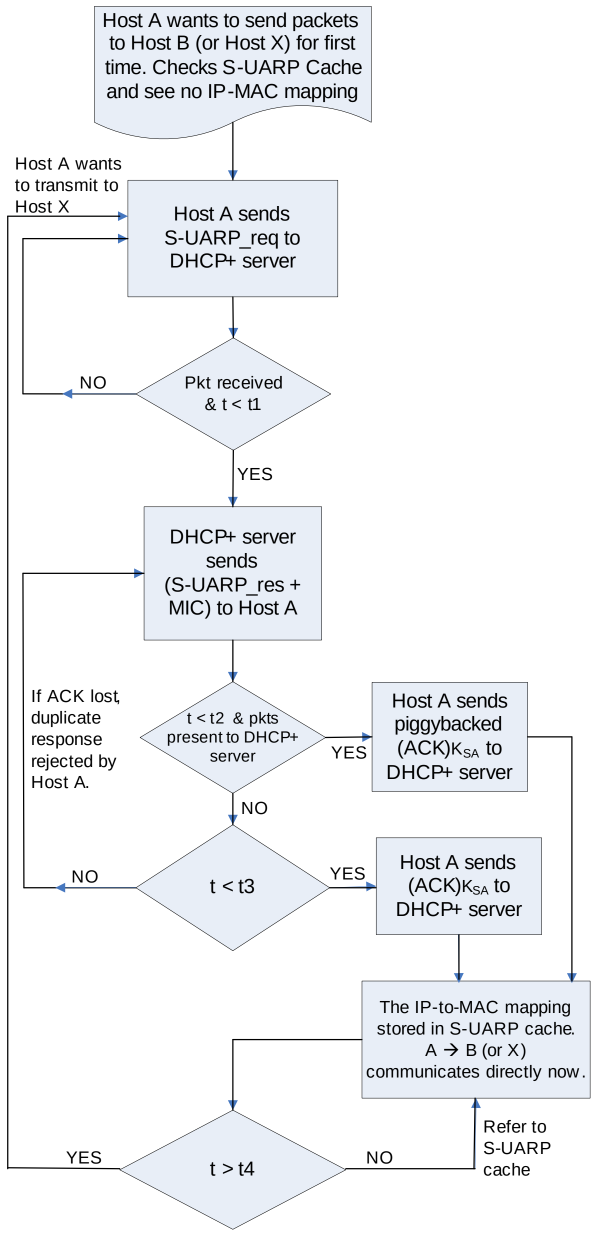
Host A wants to send packets to Host B (or Host X) for first time. Checks S-UARP Cache and see no IP-MAC mapping
Host A wants to transmit to Host X
Host A sends S-UARP_req to DHCP+ server
NO
Pkt received & t < t1
YES
DHCP+ server sends (S-UARP_res + MIC) to Host A
If ACK lost, duplicate response rejected by Host A.
t < t2 & pkts present to DHCP+ server
YES
Host A sends piggybacked (ACK)K_SA to DHCP+ server
NO
NO
t < t3
YES
Host A sends (ACK)K_SA to DHCP+ server
The IP-to-MAC mapping stored in S-UARP cache. A → B (or X) communicates directly now.
YES
t > t4
NO
Refer to S-UARP cache

Figure 3. The flowchart showing the procedure of S-UARP operation.

Note in Figure 3, t1 is the maximum time period for S-UARP response arrival (if it fails, host A would send another request), t2 is the maximum wait time for sending piggybacked ACK, t3 is the maximum acknowledgement wait time for sending ACK packet, where t3> t2 (if t > t3, the server would send response again) and t4 is the maximum wait time, until S-UARP cache is refreshed.

E. *Alternate S-UARP Protocols (with more security)*

One of the limitations of the above protocol is that the request and the response are both in clear, though this is not a serious threat considering the content of the packets. Moreover, the message integrity is only on the server's response side.

1) *Alternate Proposal 1:* A better approach needs to ensure the integrity of both S-UARP request and response as follows:

1. A → DHCP+ : S-UARP_req + MIC1
2. DHCP+ → A : S-UARP_res + MIC2
3. A → DHCP+ : (ACK, NRN) K_{SA}

In this protocol, we assume that a random number RN is known to both host and the server and is kept secret (generated by A or DHCP+). In step 1, A sends the request in clear and the MIC (i.e. MIC1). The MIC1 is generated using a collision-free one-way hash function like SHA1 that takes the secret key K_{SA},, the S-UARP_req and the random number RN as inputs. That means, MIC1 = H(K_{SA}, RN, S-UARP_req). In step 2, the server uses the S-UARP_req (in plain text), the known random number RN and secret key, K_{SA} to create a similar MIC (say, MIC1*). If MIC1 = MIC1*, then the request is accepted else it will be rejected. After verifying the integrity of the message, the server sends the response and MIC2 to the host. The MIC2 is generated in the same way (i.e. MIC2 = H(K_{SA}, RN, S-UARP_res). Finally in step 3, Host A will check the integrity of the response as in the above case (to see MIC2 = MIC2*). Host A then sends an acknowledgement and a new random number (NRN) encrypted by the secret key (K_{SA}). NRN can be used in the next request/response exchange. As in the first protocol, the acknowledgment contains the timestamp to check when the server sent the response to the host, thus protecting against replay attacks.

2) *Alternate Proposal 2:* Another more secure alternative is to use a session key S_K and an Exclusive-OR (XOR) operation as follows:

1. A → DHCP+ : S-UARP_req + MIC1
2. DHCP+ → A : S-UARP_res + $S_K \oplus$ MIC2 + MIC3
3. A → DHCP+ : MIC4

Here, MIC1 = H(K_{SA}, RN, S-UARP_req), MIC2 = H(K_{SA}, S-UARP_req, S-UARP_res), MIC3 = H(S_K, NRN), and MIC4 = H(S_K, ACK, NRN). In this protocol, the RN is generated by the server and is also known to host as a secret. In step 1, *A* sends the request and the MIC1 (using the key K_{SA}, RN and S-UARP_req). In step 2, the server checks the integrity of the message (as shown in the previous protocols), and sends S-UARP_res, $S_K \oplus$ MIC2 and MIC3 to *A*. MIC2 and MIC3 are generated using the secret key and the session key respectively. MIC2 is XORed with session key, $S_{K.}$ In step 3, host *A* checks the integrity of the message received and then compute the acknowledgment as shown in MIC4. This acknowledgement calculation involves the timestamp as in previous cases. The NRN (generated by A or DHCP+) is used by the server in MIC3 is also contained in MIC4 and is kept secret by both parties for the next request/response exchange. It is clear here that even when an attacker knows K_{SA}, he will not be able to send the acknowledgment or MIC4 as he does not know the $S_{K,}$ used. As in the previous protocol, the attacker cannot also reply an old message (replay attack) since the ACK contains the timestamp when the server generated the message in step 2. It should be noted here that in all the three protocols, both requests and responses were sent in clear to avoid extra encryption overhead. The main objective is to ensure that the message was not modified in transit and to block the possibility of an ARP poisoning by an attacker.

VI. CONCLUSION

The new S-UARP protocol is more efficient in reducing broadcast congestion in network, since the S-UARP request is unicast and directed to only DHCP+ server. The protocol is also more secure and it is quite difficult for an attacker to do ARP poisoning attack, especially on the more secure versions of S-UARP. Especially it is protected against message integrity attacks (when ARP packet content can be modified by attacker) and masquerading attacks (when new ARP bogus packet injection can be done by attacker). It should be noted that this proposal is only relevant to pervasive network based on IPv4, since ARP is implemented only in IPv4 networks. IPv6 networks use a different mechanism (called Neighbor Discovery Protocol). Nevertheless it is quite relevant until a whole conversion to IPv6 from IPv4 fully happens.

ACKNOWLEDGMENT

The author wishes to acknowledge King Fahd University of Petroleum and Minerals (KFUPM) Saudi Arabia for its support in providing the various facilities utilized in the presentation of this paper.

REFERENCES

[1] David D., Tom M., and Thad S. (2004), Mobile Phones as Computing Devices: The Viruses are Coming!, *Pervasive Computing*, pp. 11-15
[2] Roy, C. Al-Muhtadi, J. and Prasad, N.,(2002), Towards Security and Privacy for Pervasive Computing. Available: http://www.cyberdudez.com/towards-percomp-security.pdf
[3] Neal Leavitt (2000), Malicious Code Moves to Mobile Devices, *Computer Journal*, December, PP: 16-19
[4] David C. Plummer (1982) , "RFC 826 -ARP Protocol", 1982. Available: http://www.faqs.org/rfcs/rfc826.html
[5] Roney Phlips, (2007), "Securing Wireless Networks from ARP Cache Poisoning", Master's Thesis, San Jose State University, May, 2007.
[6] The TCP/IP Guide Website. Available: http://www.tcpipguide.com/free/t_ARPMessageFormat.htm
[7] Andrew S. Tanenbaum, "Computer Networks, 4e", Prentice Hall PTR, 2003, pp.450 --454.
[8] Corey Nachreiner, "Anatomy of an ARP Poisoning Attack", 2003. Available: http://www.watchguard.com/infocenter/editorial/135324.asp
[9] Bruschi, D., A. Ornaghi and E. Rosti, "S-ARP: a Secure Address Resolution Protocol", *19th Annual Computer Security Applications Conference*, 2003.