EURASIP Journal on
Wireless Communications and Networking
a SpringerOpen Journal

RESEARCH Open Access

Mitigating ARP poisoning-based man-in-the-middle attacks in wired or wireless LAN

Seung Yeob Nam*, Sirojiddin Jurayev, Seung-Sik Kim, Kwonhue Choi and Gyu Sang Choi

Abstract

In this article, an enhanced version of address resolution protocol (ARP) is proposed to prevent ARP poisoning-based man-in-the-middle (MITM) attacks in wired or wireless LAN environments. The proposed mechanism is based on the idea that when a node knows the correct MAC address for a given IP address, if it does not delete the mapping while the machine is alive, then MITM attack is not possible for that IP address. In order to prevent MITM attack even for a new IP address, we propose a new IP/MAC mapping conflict resolution mechanism based on computational puzzle and voting. Our proposed scheme can efficiently mitigate ARP poisoning-based MITM attacks, even in Wi-Fi hot-spots where wireless machines can easily come and leave, since the proposed mechanism does not require manual configuration if the proposed ARP is deployed through operating system (OS) upgrade. The proposed scheme is backward compatible with the existing ARP protocol and incrementally deployable with benefits to the upgraded machines.

1 Introduction

The address resolution protocol (ARP) is used to find the media access control (MAC) address of a node corresponding to a given IP address in the same subnet [1,2]. The resolved addresses are temporarily kept in the ARP cache to reduce the resolution time and avoid additional ARP traffic overhead for recently resolved IP addresses [3].

The ARP poisoning attack refers to the behavior of registering a false (IP, MAC) address mapping in the ARP cache of another node for malicious purposes. As an example, when there are three different nodes A, B, and C in the same subnet, if Node A registers the (IP_C, MAC_A) mapping in the ARP cache of Node B, then it is an ARP poisoning attack of Node A. If the above attack is successfully made, Node A can receive all the packets from B to C because B considers that MAC_A is the MAC address of Node C and sends all the traffic for C to MAC_A. Thus, ARP poisoning enables the attacker to eavesdrop the communication between other nodes, modify the content of the packets, and hijack the connection. This ARP poisoning can also be used to launch a denial-of-service (DoS) attack [4]. For example, if an attacker replaces the MAC address of a particular host with another value in the ARP cache of a remote machine, then the victim will experience DoS since it cannot access the original host due to the wrong MAC address. Furthermore, ARP poisoning can be used to mount man-in-the-middle (MITM) attack [5]. In the above example, if Node A registers the (IP_B, MAC_A) mapping in the ARP cache of node C additionally, then Node A can see all the packets that are exchanged between Nodes B and C. If this attack occurs, the adversary may eavesdrop, insert, or modify the messages between the victims of the MITM attack, without being detected.

If an operating system accepts any ARP reply message even though it did not send any ARP request messages, then just one transmission of the ARP reply packet will be enough to register a false (IP, MAC) mapping in the node running that operating system. Even for operating systems that do not accept unsolicited ARP reply packets, it is possible to induce an ARP request message for a specific IP address by sending a spoofed ICMP echo request packet and to register a false (IP, MAC) mapping for that IP address [6].

The ARP spoofing issue was first investigated in the environment of wired local area network (LAN), i.e., Ethernet. However, it is becoming easier to access Internet via wireless LAN, i.e., 802.11 network, due to the widespread deployment of access points (APs) by many

* Correspondence: synam@ynu.ac.kr
Department of Information and Communication Engineering, Yeungnam University, Gyeongsan-si, Gyeongbuk 712-749, Korea

Internet service providers (ISPs). Thus, ARP spoofing can also be an important issue in the wireless LAN environment. According to [7], there can be many scenarios whereby a wireless attacker poisons the ARP caches of other wireless nodes, connected through the same AP or different APs, or it poisons the ARP caches of other wired nodes in the same subnet. Although there have been several attempts to resolve the ARP cache poisoning problem in the wired LAN environment, some of those approaches may not be effective in a different environment of wireless LAN.

Conventional approaches can be classified into two categories depending on whether the ethernet switch upgrade is required or not. dynamic ARP inspection (DAI) [8] requires the support from the ethernet switches. DAI usually requires manual configuration by network managers and cannot prevent ARP poisoning occurring between wireless nodes connected through the same AP. The methods that do not require support from Ethernet switches usually use cryptographic techniques [9,10]. Since most of these methods use public key cryptography, there should be a central server to distribute the public key of each node reliably. However, this central server could be subject to a single point of failure problem.

Furthermore, the public key of the central server itself and the MAC address of the server need to be delivered to each node reliably. This step usually requires manual setting by the network manager, and it may not be appropriate for the environment of Wi-Fi hot-spots where wireless machines can easily come and leave.

Thus, we investigate a new mechanism to prevent ARP spoofing-based MITM attacks in wired or wireless LAN environments, while overcoming the limitations of existing approaches. The proposed scheme makes the upgraded normal nodes protect each other through collaboration based on voting and computational puzzles. Since the public key cryptography is not required, the manual setup or configuration is not required for the newly arriving wireless nodes and it is free from a single point of failure problem due to the absence of any central server. The remainder of the article is organized as follows. We first discuss related study in Section 2. Section 3 describes the proposed enhanced version of ARP. In Section 4, the performance of the proposed mechanism is verified through experiments in a testbed environment. Finally, conclusions are presented in Section 5.

2 Related study

As explained in Section 1, neglecting unsolicited ARP replies cannot prevent ARP cache poisoning, since ARP requests can be easily induced by source address-spoofed ICMP echo request messages. If the member nodes of a given subnet do not change frequently, then the ARP cache poisoning might be avoided by employing static ARP. However, this approach may be infeasible in an environment where the IP addresses are allocated to mobile nodes dynamically through Dynamic Host Configuration Protocol (DHCP). DAI corresponds to the approach that requires the support of ethernet switches [8]. In the DAI, the ethernet switch checks the validity of the received ARP packet based on the trusted IP-to-MAC mapping database. However, this database is either manually managed or dynamically managed through DHCP snooping [8]. In order to perform DHCP snooping, the port on which the network DHCP server is connected needs to be configured as a *trusted* interface, and other ports need to be configured as *untrusted* interfaces. However, this approach may not be effective, if the ARP cache poisoning occurs among the wireless nodes connected via the same AP. In this article, we attempt to devise a method to prevent ARP poisoning-based MITM attacks with minimal overhead of manual configuration by the network administrator or by the user, and with minimal infrastructure upgrade cost.

The approaches that do not require the upgrade of ethernet switches can be classified into two sub-categories based on the use of cryptography. Antidote [11] is a non-cryptographic approach. If a new ARP reply, which is insisting on a new MAC address for an existing IP address, arrives, the new (IP, MAC) mapping is accepted, only when the node corresponding to the previous MAC address is not alive. However, there is a problem in Antidote. When a normal machine sends an ARP request for an IP address which is not in the ARP cache currently, if a malicious ARP reply arrives first, then the victim caches the wrong reply and discards the later ones. Thus, MITM attack can be successful under this race condition. The proposed scheme shares the basic concept, preservation of (IP, MAC) mapping while the node corresponding to the original MAC address is alive, with the Antidote, but resolves the race condition in a different way using computational puzzle-based voting.

S-ARP [9] is a popular cryptography-based approach. Although S-ARP requests operate in the same way as normal ARP requests, S-ARP replies need to be signed by the sender's private key and the signature needs to be verified using the sender's public key at the receiver side. Since S-ARP is based on asymmetric cryptography, it requires a central server called authoritative key distributor (AKD) to manage the mapping between the IP address and the corresponding public key. If the AKD fails, the whole system may not work, i.e., S-ARP has a single point of failure problem. In addition, S-ARP usually requires the upgrade of the DHCP server and incremental deployment is not easy, since the S-ARP-enabled machine may not accept ARP replies from non-S-ARP nodes.

Goyal and Tripathy proposed a modified version of ARP [6] to lower the computational cost of S-ARP. This avoids the calculation of digital signature for each ARP reply by allowing the use of the same digital signature several times over multiple ARP replies during a pre-defined time period, while ensuring the recency of the mapping in the digital signature through one time passwords. Although the computational cost is relieved, it has the same limitation as S-ARP: a single point of failure problem due to AKD, and the cost of manual configuration to disseminate the public key and the MAC address of AKD to all newly arriving hosts.

Ticket-based ARP (TARP) [10] is another approach to reduce the computational cost of S-ARP by employing the concept of ticket, centrally generated IP/MAC address mapping attestation. However, TARP also requires manual dissemination of Local Ticket Agent (LTA)'s public key, and LTA might be subject to a single point of failure problem.

Philip [12] proposed an approach to prevent ARP cache poisoning in wireless LAN by implementing the defense mechanism in the AP. The basic idea is as follows. The AP constructs the list of correct IP-to-MAC address mapping by monitoring DHCP ACK messages or referring to the DHCP leases file, and blocks all the ARP packets with a false mapping based on the constructed list. However, this approach can be applied only to the dynamic IP addresses allocated through DHCP, and cannot prevent ARP cache poisoning occurring inside the wired LAN. We attempt to protect the upgraded nodes from ARP poisoning-based MITM attacks, whether those nodes are connected to LAN through wire or wireless medium.

Recently, Nam et al. [13] proposed an enhanced version of ARP, called MR-ARP, to prevent ARP poisoning-based MITM attacks in the ethernet by employing the concept of voting. If an ARP request or reply message arrives declaring a new MAC address for an IP address registered in the ARP cache, MR-ARP queries the node corresponding to the current MAC address to check if that IP address is still used by that node in a similar way to Antidote. If an MR-ARP node receives an ARP request or reply declaring an (IP, MAC) mapping for a new IP address, it requests that the neighbor nodes vote for the new IP address to make the correct decision for the corresponding MAC address. For this mechanism, the voting can be fair only when the voting traffic rates of the responding nodes are almost the same. This condition can be satisfied in the Ethernet, but may not be valid in the 802.11 network due to the traffic rate adaptation based on the signal-to-noise ratio (SNR), i.e., auto rate fallback (ARF) [14,15]. Thus, we attempt to overcome the limitation of MR-ARP by improving the voting procedure through the incorporation of computational puzzles, while achieving the following goals: mitigation of ARP poisoning-based MITM attacks in wired or wireless LAN, backward compatibility with existing ARP, minimal infrastructure upgrade cost (e.g., no upgrade of LAN switches or modification of DHCP), incremental deployability, and no manual configuration for newly joining nodes.

3 Computational puzzle-based enhanced ARP

We improve the voting procedure of MR-ARP by incorporating computational puzzles [16-18] to mitigate ARP poisoning-based MITM attacks in wired or wireless LAN, while overcoming the limitation of MR-ARP [13]. As explained at the end of Section 2, the transmission rates of wireless nodes may not be uniform, and thus, the fair voting may not be guaranteed by the voting mechanism of existing MR-ARP. The key idea of the new voting mechanism to resolve the fairness issue in the wireless LAN can described as follows. If the CPU processing power of different nodes is not significantly different, then they might spend a similar amount of time solving the puzzles of the same difficulty. Thus, if we accept the votes only from the nodes that solved the puzzle correctly, then fair voting might be realized in the wireless environment, too. Furthermore, voting reply traffic can be reduced compared to the original MR-ARP scheme since we no longer need to receive multiple votes from each neighbor node. In addition to the fairness issue, MR-ARP could be vulnerable to ARP cache-poisoning problem, although not the MITM problem, especially when a new machine with a new IP address is added. We also explain how this issue is resolved through the new initialization stage after the detailed description of the new voting mechanism. The proposed enhanced version of MR-ARP, which is called EMR-ARP, follows a similar approach to the original MR-ARP, except in the voting and initialization procedures. Thus, EMR-ARP is also based on the following idea [13]. When Node A knows the correct IP/MAC address mapping for Node B, if Node A retains the mapping while Node B is alive, then ARP poisoning and the MITM attack between A and B are not possible.

EMR-ARP retains the long-term (IP, MAC) mapping table, in addition to the normal ARP cache, which is also referred to as the short-term table in this article, to manage the (IP, MAC) mapping for all alive machines in the subnet. Three fields, IP, MAC, and Timer T_L, are allocated for each IP address registered in the long-term table. The default value of the timer in the long-term table is 60 minutes. In order to avoid losing the mapping of (IP_a, MAC_a) for an alive host after 60 minutes, the EMR-ARP node sends new ARP request messages for IP_a only to MAC_a through unicasting and checks whether MAC_a is alive and is still using IP_a just before the timer expires. In this case, 10 ARP messages are sent at the random intervals of 50~100 ms. If at least one ARP reply returns, then the mapping is registered in the short-term ARP cache and the corresponding long-term table timer is set to 60 min again. If there is no ARP reply, then the mapping between IP_a and MAC_a is considered invalid and the corresponding entry is deleted from the long-term table. Thus, the (IP, MAC) mapping can be retained in the long-term table until the binding is released.

EMR-ARP inspects the mapping between the sender protocol (IP) address and the sender hardware (MAC) address of every incoming ARP request message to track the IP/MAC mapping for all alive machines in the same subnet, since each alive machine tends to send ARP requests to find the MAC address of the gateway router repeatedly due to the periodic timer expiry in the ARP cache. However, IP/MAC mapping of every ARP request cannot be directly reflected in the long-term table due to the possibility of ARP cache poisoning attempts. Thus, the short-term cache, i.e., ARP cache, and long-term table need to be updated carefully on arrival of ARP request or reply packets. Figure 1 shows the detailed short-term cache and long-term table management policy. By the rule for the case (A) in Figure 1, each IP/MAC mapping registered in short-term cache is frozen until it expires, e.g., for 2 minutes for Windows XP, to prevent too frequent cache updates by ARP request sniffing.

In case (B) of Figure 1, MAC conflict occurs, because the newly received MAC address MAC_a for IP_a differs from MAC'_a that is already associated with IP_a. This conflict is resolved by giving a priority to MAC'_a only if it is alive. As shown in Figure 1, the activity of a host is examined by sending 10 ARP request packets and counting the ARP replies. Multiple ARP requests are sent to cope with unexpected packet losses, including the case of DoS attack on MAC'_a. Even though the packet loss probability is as high as 80% by DoS attack, at least one ARP reply will be returned with a probability of 89%(= $1 - 0.8^{10}$). Let us consider the case where the mapping of (IP_a, MAC_z) expires, and the IP address IP_a is allocated to a new machine with the MAC address of MAC_a through DHCP. In this case, the new mapping between IP_a and MAC_a can be chosen by the rule corresponding to the case (B) in Figure 1, even though the mapping (IP_a, MAC_z) exists in the long-term table. When a first ARP request from MAC_a arrives, there would be conflict on IP_a in the long-term table, and the node which observes the conflict will send ten ARP requests to MAC_z. However, the machine MAC_z would not reply and the new mapping can be registered, since the previous mapping has expired.

```
/* (IP_a, MAC_a): the sender protocol (IP) and hardware (MAC)
                  addresses of the received ARP request or reply packet */

if(IP_a is registered in the short-term cache) { --- (A)
  no action; /* in case of conflict, preserve existing mapping.*/
}
else if((IP_a, MAC_a) is registered in the long-term table) {
  register (IP_a, MAC_a) in the short-term cache;
  set the long-term table timer to 60 minutes;
}
else if(IP_a is in the long-term table, but the registered MAC is not MAC_a) { --- (B)
  /* conflict on IP and MAC mapping */
  send 10 ARP requests to existing MAC through unicasting
    at random intervals with an average of 10 msec;
  if(at least one ARP reply arrives)
    retain the existing (IP, MAC) mapping and drop the new one;
  else
    accept the new mapping;
  The accepted mapping is registered in the short-term cache, too.
}
else{ /*i.e. IP_a is not in short-term/long-term tables */ --- (C)
  send voting requests for IP_a;
  if(no response)
    the mapping (IP_a, MAC_a) is registered in both tables;
  else if(there exists a MAC that polls over 50% of votes for IP_a)
    that mapping is registered in both tables;
}
```

Figure 1 Short-term cache and long-term table update policy applied on the arrival of ARP request or reply packets.

3.1 Computational puzzle-based voting scheme

Thus far, we investigated how to prevent MITM attacks for the IP addresses whose corresponding MAC addresses are known already. However, if Node A receives an ARP request from a new IP address, then Node A cannot easily know the correctness of the source IP/MAC mapping contained in the ARP request. For example, when a new machine is added into a LAN with empty short-term cache and long-term table and the machine sends an ARP request for the gateway router, if an adversary's ARP reply advertising a fake MAC address arrives first, then this wrong MAC address may be registered and the late true MAC address may be dropped. We use computational puzzle-based voting, which corresponds to the case (C) in Figure 1, to solve the poisoning problem for this particular case.

The basic idea of the voting-based resolution mechanism can be explained as follows. When Node A is turned on with empty ARP cache and long-term table, if the neighbor nodes that know the true MAC address of the gateway router provide that information to Node A, then Node A can make a correct decision on the MAC address of the gateway router based on the majority of the votes. Figure 2 describes a simplified outline of the voting procedure with this example scenario. Since Node A wants to know the true MAC address of the gateway router, it first generates a new puzzle and broadcasts the voting request message containing the new puzzle to neighbor EMR-ARP nodes. Then, the neighbor nodes solve the puzzle using the message transformation function, and notify the voting request node of the puzzle solution and the MAC address of the gateway router that they know through voting reply messages. After collecting the voting reply messages from the neighbor nodes, Node A verifies the puzzle solution using the message recovery function, and makes a decision based on the votes from the nodes providing the correct puzzle solution. The message transformation function and the message recovery function will be described shortly in this section.

Preserving the fairness among different nodes is one of the most important issues in the voting scheme. In the previous version of the voting scheme used in MR-ARP [13], the fairness was provided by having all the neighbor MR-ARP nodes send multiple voting reply packets at the maximum speed of link rate in the Ethernet. This idea works, because the LAN cards usually have sufficient capability to send traffic at the link rate regardless of the vendor. However, the maximum transmission rates of different nodes may not be the same in the wireless LAN if the transmission rate changes by ARF depending on SNR. Thus, the voting scheme needs to be refined to provide the fairness under any combination of wireless and wired nodes. We attempt to provide the fairness by having the wired or wireless nodes spend a similar amount of time solving the computational puzzles [18] of the same difficulty and making the address resolution decision only based on the votes from the

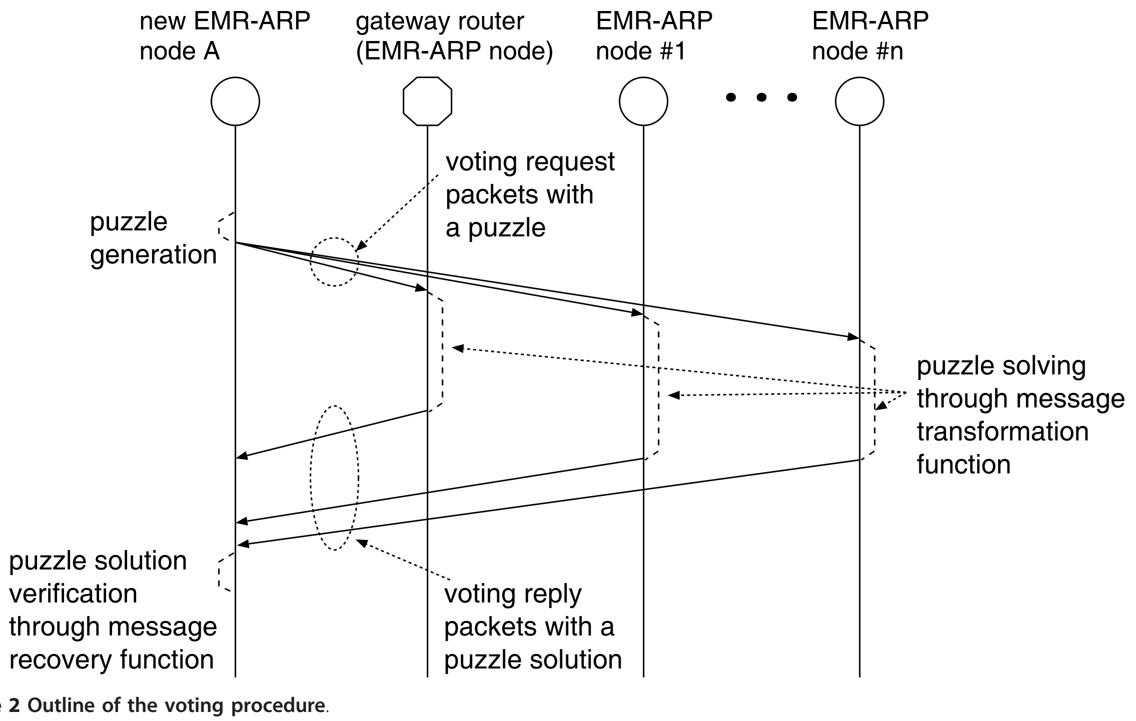
new EMR-ARP node A
gateway router (EMR-ARP node)
EMR-ARP node #1
EMR-ARP node #n
voting request packets with a puzzle
puzzle generation
puzzle solving through message transformation function
puzzle solution verification through message recovery function
voting reply packets with a puzzle solution

Figure 2 Outline of the voting procedure.

nodes that solved the puzzle correctly. Thus, the puzzle used in the proposed ARP scheme needs to satisfy the following requirements.

- The puzzle must be easy to make.
- The puzzle must be difficult to solve, but it should be easier to verify the puzzle solution.
- Puzzle receiver nodes cannot predict the puzzle easily.
- Different nodes should solve different problems to avoid the case whereby a malicious node pretends to be many other nodes by sending multiple replies with a spoofed MAC address after solving only one problem.
- The complexity of the different puzzles should be similar to provide fairness between different nodes.

Conventional computational puzzles [16-18] based on hash functions might satisfy most of the above requirements. However, we found very large variance in the puzzle solving time for hash function-based puzzles. As an example, let us consider the puzzle that finds the value of x that satisfies the following relation:

$$[H'(x)]_r = [y]_r,$$

where H' is a uniform random hash function, r is a positive integer, y is an integer given by the puzzle presenter, and $[z]_r$ means the least significant r bits of z. Then, the number of iterations required to find a solution of the above puzzle can be modeled as a random variable X that has a geometric distribution with a success probability p, where $p = 1/(2^r)$. The mean (μ) and the standard deviation (σ) of X are $1/p$ and $\sqrt{1-p}/p$, respectively. Thus, σ can be large, even close to μ for small values of p. When this type of puzzle is used, if the attacker can luckily solve many puzzles, even for different MAC addresses, while most other normal machines solve only one puzzle, then the attacker might win the voting by increasing the ratio of votes supporting the fake MAC address and spoof the ARP cache of the voting request node. The variance in the puzzle solving time needs to be minimized to avoid such a problem.

In order to satisfy the above requirements, while minimizing the variance in the puzzle-solving time, we design the puzzle based on the public key cryptography, especially RSA algorithm [19,20], so that the number of arithmetic operations required to solve the puzzle cannot differ depending on the machines. The proposed puzzle can be described as follows. We first consider two keys e and d that are usually termed the public key and the private key in the public key cryptography. In the proposed scheme, the value of e is fixed to 3, a popular value for the public key [20]. We calculate the value of d in the following way. We choose two different large primes p and q, that are 128 bits long and satisfy the condition that (p - 1) and (q-1) are relatively prime to e. Then, d is an integer that satisfies the condition:

ed = 1 mod (p - 1)(q - 1). Such a value of d can be easily found with Euclid's algorithm [20]. N is defined as $N = pq$, and N is 256 bits long. When $e = 3$, if the message x is smaller than $\sqrt[3]{N}$, then the encrypted message x^3 can be decrypted simply by taking a cube root. This problem can be avoided by padding each message with a random number before encryption, so that the padded message is sufficiently large. We define the message transformation functions $f_i(i = 1,..., m)$, and F_m as follows.

$$\begin{aligned} f_i(x) &= \{(x^d \bmod N) - 1 + i\} \bmod (N-1) + 1, \\ F_m(x) &= f_m(f_{m-1}(\cdots f_1(x))), \end{aligned} \quad (1)$$

If f_i is simply defined as modular exponentiation, e.g., as the decryption operation in the public key cryptography, without any additional operations of addition and subtraction, then the multiple encryption required to evaluate $F_m(x)$ can be simplified into a single exponentiation by Euler's theorem [20]. In order to avoid such a simplification, we incorporated the additional operations of addition and subtraction in the definition of the message transformation functions as shown above.

We also define the message recovery functions $g_i(i = 1,..., m)$, and G_m as follows.

$$\begin{aligned} g_i(x) &= \{(x - 1 - i) \bmod (N-1) + 1\}^e \bmod N, \\ G_m(x) &= g_1(g_2(\cdots g_m(x))). \end{aligned} \quad (2)$$

Then, from (1) and (2) we can obtain the following relation.

$$\begin{aligned} g_i(f_i(x)) &= \{(f_i(x) - 1 - i) \bmod (N-1) + 1\}^e \bmod N, \\ &= [\{((x^d \bmod N) - 1 + i) \bmod (N-1) - i\} \\ &\quad \bmod (N-1) + 1]^e \bmod N. \end{aligned}$$

If we put $y = (x^d \bmod N)$-1, then y belongs to [0, N - 2] and the above relation can be changed into

$$\begin{aligned} g_i(f_i(x)) &= \{y \bmod (N-1) + 1\}^e \bmod N, \\ &= (y+1)^e \bmod N, \\ &= (x^d \bmod N)^e \bmod N, \\ &= x, \end{aligned}$$

where the last equality is valid by the RSA algorithm. Similarly, we can show that

$$G_m(F_m(x)) = x, \quad \text{if } x < N. \quad (3)$$

The proposed puzzle is to find the value of y that satisfies the following relation for a given integer x,

$$y = F_m(x).$$

Thus, if the values of e, d, N, and m are fixed, then the puzzle complexity in terms of the number of operations is also fixed. In the proposed scheme, the values of e, d, and N are fixed as described below, and the complexity of the puzzle is controlled only by m.

In EMR-ARP, we use two types of puzzles. The first type of puzzle should be solved by the voting request node itself to prevent a DoS attack that attempts to exhaust the processing power of neighbor EMR-ARP nodes by inducing continuous puzzle computation through too frequent transmission of voting request packets. The second type of puzzle needs to be solved by the EMR-ARP nodes receiving the voting request packets. This puzzle is used to provide the fairness among different nodes in voting.

In the first-type puzzle, the voting request node first makes a message m_s by concatenating its own MAC address ($\mathrm{MAC_A}$) and the local time T_s measured in seconds when the voting request node starts to calculate the puzzle, i.e., $m_s = \mathrm{MAC_A}||T_s$, and transform m_s into c_s by the message transformation function F_m. Then, the voting request node sends T_s, m, and the transformed message c_s to the neighbor nodes. e, d, and N need not be delivered since they are fixed and known to EMR-ARP nodes in advance. The neighbor nodes apply the message recovery function G_m to the received message c_s. If the recovered message agrees with $\mathrm{MAC_A}||T_s$, then it confirms that the sender solved the first-type puzzle correctly and the receiver proceeds to solve the second-type puzzle to send a valid vote in return.

The second puzzle can be described as follows. The voting request node with the MAC address of $\mathrm{MAC_A}$ generates the key parameter for the second puzzle P by

$$P = H(K \,\|\mathrm{MAC_A}\|\, T_s), \qquad (4)$$

where H is a publicly-known cryptographic hash function, and MD5 is used for H in the current version. K is the secret that is known only to the voting request node. If the value of P is delivered to the neighbor nodes additionally, then each receiver node generates a message m_r by $m_r = \mathrm{MAC_A}||\mathrm{MAC_R}||P$, where $\mathrm{MAC_R}$ is the MAC address of the receiver node. We make each EMR-ARP node solve a puzzle that differs from those of other nodes by including the MAC address of the receiver inside the puzzle to prevent the reuse of the solution. m_r is transformed by F_m into c_r, and only the transformed message c_r is sent back to the voting request node. The voting request node applies the recovery function G_m to the received message c_r, and compares the recovered message with m_r, whose components are known to voting request node already. If c_r is correctly calculated, i.e., $c_r = F_m(m_r)$, then the following relation should hold by (3):

$$G_m(c_r) = m_r.$$

Thus, by comparing $G_m(c_r)$ and m_r, we can verify the correctness of each answer. We can easily find that the second-type puzzle reflects all the requirements stated above.

The implementation-related details are as follows. Two more ARP packet types, voting request and voting reply packets, are defined in the EMR-ARP in a similar way to MR-ARP. The *operation* field is set to 20 and 21 for voting request and reply packets, respectively. Figure 3 shows the packet formats for voting request and reply messages. As shown in the figure, the voting request packet is defined by adding 4 more fields, (T_s, m, c_s, P), to the normal ARP request/response packet format, and the voting reply packet is defined by adding just one more field of c_r.

Each EMR-ARP node maintains a table of (MAC address of voting request node, T_s) pairs, called *puzzle history table*, to prevent the flooding of the identical puzzles. T_s is the puzzle generation time at the sender node as described above. If an EMR-ARP machine receives a voting request packet, then it first examines the value of the T_s field. If the value of received T_s is less than or equal to the value stored in the puzzle history table, then the received voting request message is neglected since the puzzle is highly likely to be a repetition of an old one. After verifying the validity of T_s to avoid the flooding of identical puzzles, the EMR-ARP node investigates the correctness of the solution contained in the field of c_s. If the solution is correct, the received voting request is considered valid, and the received timestamp T_s is registered in the puzzle history table, i.e., in the entry corresponding to the MAC address of the current voting request node. Then, the EMR-ARP node proceeds to solve the second-type puzzle to provide a feedback for the voting request.

After solving the second-type puzzle as described above, the EMR-ARP node sends the MAC address corresponding to the queried IP address and the puzzle solution to the voting request node through the fields of target hardware address and c_r, respectively, in the voting reply packet of Figure 3b.

Then, the voting request node receives and buffers the incoming voting reply packets up to 1 s from the voting request transmission time. The voting request node finds the earliest voting reply packet, calculates the response time t_1, i.e., the period from the voting request transmission time to the arrival time of the earliest

Hardware Type		Protocol Type
Hardware length	Protocol length	Operation
Sender Hardware Address (bytes 0-3)		
Sender Hardware Address (bytes 4-5)		Sender Protocol Address (bytes 0-1)
Sender Protocol Address (bytes 2-3)		Target Hardware Address (bytes 0-1)
Target Hardware Address (bytes 2-5)		
Target Protocol Address (bytes 0-3)		
T_s		
n		
d		
c_s		
P		

(a) Voting request packet format

Hardware Type		Protocol Type
Hardware length	Protocol length	Operation
Sender Hardware Address (bytes 0-3)		
Sender Hardware Address (bytes 4-5)		Sender Protocol Address (bytes 0-1)
Sender Protocol Address (bytes 2-3)		Target Hardware Address (bytes 0-1)
Target Hardware Address (bytes 2-5)		
Target Protocol Address (bytes 0-3)		
c_r		

(b) Voting reply packet format

Figure 3 Voting request and reply packet formats for EMR-ARP. (a) Voting request packet format, (b) Voting reply packet format.

voting reply packet, and discards the voting reply packets that arrive later than $2t_1$ since those late replies might come from the malicious nodes actively solving the puzzles multiple times even for different nodes to increase the ratio of the votes supporting the fake IP/MAC mapping. Then, the voting request node verifies the correctness of the puzzle solution only for the not-too-late replies and accepts the received IP/MAC mapping only when the solution for the second-type puzzle is correct. The voting request node calculates the polling score for each candidate MAC address. If a MAC exists that received over 50% of valid votes, then that MAC address is accepted for the queried IP address.

3.2 Initialization stage

The original MR-ARP scheme might be vulnerable especially when a new MR-ARP node is attached to a LAN where an attacker resides. The detailed problem is described with an example below. In this section, we introduce an initialization stage for the proposed EMR-ARP to overcome the shortcoming of the original MR-ARP scheme especially for this case.

Figure 4 shows an example network topology to describe the vulnerability of the original MR-ARP scheme. In the figure, Nodes A, B, C, and D are MR-ARP enabled nodes. Among them, D is a router connecting the given subnet to other subnets. We assume that there is only one attacker, i.e., Node E, and Node F is newly attached to the LAN. Let us assume that Node F is also MR-ARP enabled and uses a new IP address not known to other MR-ARP nodes. In this case, if the new node sends an ARP request message querying the MAC address of the router, then Node F might receive a false IP/MAC mapping from Node E. However, the correct MAC address can be resolved through the voting procedure since the number of MR-ARP nodes is larger than that of the attackers.

In this scenario, existing MR-ARP nodes do not know the IP address of Node F, and thus, each of them will try to find the correct MAC address of Node F through voting after waiting a random time between 0 and 100 ms [13]. If we assume Node D is the first node that queries the MAC address of Node F, Node D will send a voting request for the IP address of Node F, IP_F. There are one attacker node, i.e., Node E, and only one MR-ARP node that knows the MAC address for IP_F, i.e., Node F itself. If those two nodes receive the voting request for IP_F, then Node F will reply with 50 voting reply packets containing the correct MAC address for

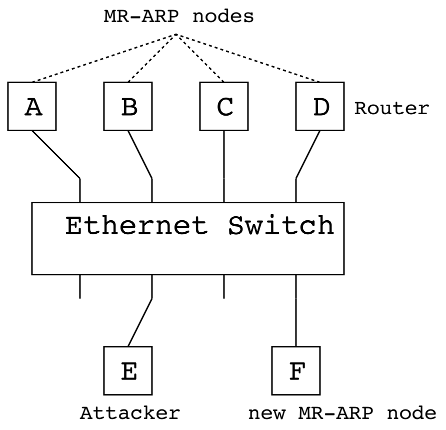

Figure 4 Example network topology to illustrate one problem of the original MR-ARP that happens when a new MR-ARP node is attached.

IP_F. The attacking node needs not follow the original MR-ARP protocol, and it can send more than 50 voting reply packets with a false IP/MAC mapping. Thus, the attacker can win the voting in this case.

MITM attack cannot be successful between Nodes D and F even in this case since the ARP cache and the long term table of Node F are protected with the help of other MR-ARP nodes. However, we further investigate how to prevent ARP cache poisoning-based one-way eavesdropping by Node E for the packets going from the router Node D to the new Node F.

We employ additional initialization steps for EMR-ARP, which are described hereafter, to resolve this problem. The initialization procedure is performed only once during the lifetime of a given machine. The lifetime means the time interval from the time when the machine is turned on to the time when the machine is turned off. If a given EMR-ARP machine receives a new IP address through DHCP, then it performs the initialization steps, immediately after the IP address is configured. If the IP address is manually configured for a given EMR-ARP machine, then the initialization steps are performed just after the network card is activated. We need to note that all the ARP packets irrelevant to the initialization steps are discarded until the initialization steps are completed.

The initialization stage consists of two steps. In the first step, the rebooted or newly attached node advertises its own IP/MAC mapping through a gratuitous ARP request packet [21]. Each EMR-ARP node that receives the gratuitous ARP packet checks whether it knows the source IP address of the received packet. If it knows the received IP address, then it resolves the conflict in the usual way, by asking the previous owner of that IP address and giving it a priority. If the EMR-ARP node does not know the received IP address, then it just neglects the received gratuitous packet. Figure 5 summarizes the task that is performed by the EMR-ARP node receiving a gratuitous ARP request packet. This packet is sent first to update the IP/MAC mapping for the given IP easily, without the burden of voting especially when the owner of a given IP address changes legitimately through DHCP.

The second step starts after waiting 1 s from the transmission of the gratuitous ARP request packet. In the second step, the rebooted or newly attached node sends a special voting request packet for its own IP address to know if its current IP address is used by other machines. If there is no voting reply packet, or the MAC address of voting node polls over 50% of votes, then the new node can use the current IP address without any problem. If there is at least one reply packet and the MAC address of voting request node polls less than 50%, then the node gives up the use of the current IP address and requests a new IP address through DHCP or manual configuration with the help of a network administrator.

If other existing EMR-ARP nodes retain an entry corresponding to the IP address of the voting request node in the long term table, then they first verify the correctness of the solution to the first-type puzzle and send voting reply packets after solving the second-type puzzle, only if the first puzzle solution is correct. Although the voting reply packets for a normal voting request message are sent only to the voting request node through unicasting, the voting reply packets for a special voting request message are broadcasted, so that other neighbor EMR-ARP nodes can sniff those responses.

If an existing EMR-ARP node does not know the IP address of the voting request node, then the node temporarily stores the sender protocol and sender hardware address mapping of the received ARP voting request

```
/* (IP_a, MAC_a): the sender protocol (IP) and hardware (MAC)
                  addresses of the received gratuitous ARP request packet */

if((IP_a, MAC_a) is registered in the long-term table) {
  register (IP_a, MAC_a) in the short-term cache;
  set the long-term table timer to 60 minutes;
}
else if(IP_a is in the long-term table, but the registered MAC is not MAC_a) {
  /* conflict on IP and MAC mapping */
  send 10 ARP requests to existing MAC through unicasting
    at random intervals with an average of 10 msec;
  if(at least one ARP reply arrives)
    retain the existing (IP, MAC) mapping and drop the new one;
  else
    accept the new mapping;
  The accepted mapping is registered in the short-term cache, too.
}
```

Figure 5 Short-term cache and long-term table update policy applied on the arrival of gratuitous ARP request packets.

packet. The neighbor EMR-ARP node ignorant of the IP address of the node sending a special voting request packet monitors the reply packets from other EMR-ARP nodes for an interval of 1 sec. If there is no ARP voting reply during that time interval, then the neighbor EMR-ARP node accepts the sender protocol and sender hardware address mapping of the special voting request packet into the ARP cache and the long term table. However, the mapping is not accepted, if there is any voting reply packet. Figure 6 summarizes the way in which EMR-ARP nodes process a special voting request message that is querying the MAC address for the IP address of the voting request node.

Thus, according to the initialization procedure described above, when an EMR-ARP node sends the first special voting request packet after reboot or initial deployment, if an attacker replies with a false MAC address for the queried IP address, then the new node will try to avoid the use of the IP address in conflict. Thus, the voting replies from the attackers advertising false IP/MAC mapping cannot help one-way eavesdropping for the packets going to the new EMR-ARP node. Conversely, if there is no reply for the first special voting request, then all the normal EMR-ARP nodes will accept the mapping between the sender protocol address and the sender hardware address of the special voting request packet by the algorithm in Figure 6. Thus, the proposed initialization procedure can effectively prevent one-way eavesdropping of the packets destined to the newly attached nodes.

4 Numerical results

In this section, we evaluate the performance of the proposed ARP in terms of the voting traffic overhead, the reliability in the presence of attackers, and address conflict resolution delay.

4.1 Analysis of traffic overhead

We first compare the traffic overhead of EMR-ARP with that of MR-ARP. The traffic overhead of EMR-ARP or MR-ARP is due to the exchange of voting request and reply packets in the voting procedure. Thus, we investigate the traffic rate of voting request and reply packets.

We consider an example case with several assumptions to simplify the analysis. We assume that IP addresses are allocated through DHCP for all nodes in the same subnet, the IP lease time is fixed to T_D, and a limited set of IP addresses are reused through DHCP with a minimal duration of IP address reassignment so that once an IP address is registered in the long-term table, it is not evicted from the table. We assume that short-term and long-term tables of a given machine are cleared when a new IP address is assigned to that machine to consider a situation with high voting traffic overhead. We also assume that there is no attacker in the subnet to focus on the traffic overhead under normal conditions. We first analyze the traffic overhead for MR-ARP. Let L and M denote the total number of alive machines and the total number of alive MR-ARP machines in the LAN, respectively. When machine A restarts with an empty short-term Cache and long-term table, Node A attempts to find the MAC addresses corresponding to the observed IP addresses through the voting procedure described in [13]. When Node A sends a voting request packet for IP address IP_B, the average rate of the voting reply traffic to Node A becomes $(M - 1) \times 50 \times 28 \times 8/T_D = 11.2(M - 1)/T_D$ Kbps. Thus, the aggregate voting reply rate to Node A, $r_A^{\text{MR - ARP}}$, becomes

```
/* (IP_a, MAC_a): the sender protocol (IP) and hardware (MAC)
                   addresses of the received voting request packet,
   IP_g: the target protocol (IP) address of the received packet */

if(IP_g is registered in the long-term table) {
  check the correctness of the first puzzle solution;
  solve the second puzzle and reply by broadcasting the MAC address
    for IP_g only when the first puzzle solution is correct;
}
else {
  temporarily store the mapping of (IP_a, MAC_a),
    and monitor voting replies from other EMR-ARP nodes for 1 sec;
  if(no reply for 1 sec)
    accept the mapping of (IP_a, MAC_a) into short and long term tables;
  else
    drop the mapping;
}
```

Figure 6 Handling of the received special voting request packet whose sender protocol (IP) address is equal to the target protocol (IP) address.

$11.2(L - 1)(M - 1)/T_D$ Kbps. Similarly, the aggregate voting request rate to Node A can be expressed as $(M - 1) \times (L - 1) \times 28 \times 8/T_D$. Thus, the average voting traffic rate to Node A can be expressed as

$$\begin{aligned} r_A^{\text{MR - ARP}} &= \frac{11.2(L-1)(M-1)}{T_D} + \frac{0.224(L-1)(M-1)}{T_D} \\ &= \frac{11.4(L-1)(M-1)}{T_D}\text{Kbps.} \end{aligned} \tag{5}$$

We now analyze the traffic overhead for EMR-ARP nodes under the same assumptions. Although both MR-ARP and EMR-ARP use the voting request and reply packets, the detailed formats of these packets differ. MR-ARP reuses the packet format of ARP request or reply messages for the voting request and reply messages without any modification. However, the voting request and reply packet formats in EMR-ARP are defined by adding a few additional fields, (T_s, m, c_s, P) for the voting request packet and c_r for the voting reply packet, to the packet format of ARP request/reply message. In the current implementation, the sizes of these additional fields are set as follows: T_s is 4 bytes, m is 4 bytes, c_s is 32 bytes, P is 16 bytes, and c_r is 32 bytes. Thus, the sizes of the voting request and reply packets are 84 and 60 bytes, respectively, in EMR-ARP, which are larger than 28 bytes in MR-ARP. Other major differences between EMR-ARP and MR-ARP are as follows. Normal MR-ARP node responds to a single voting request packet by sending 50 voting reply messages, but a benign EMR-ARP node replies with a single voting reply message. Under the above assumptions, the voting traffic rate to EMR-ARP Node A, $r_A^{\text{EMR - ARP}}$, can be expressed as

$$\begin{aligned} r_A^{\text{EMR - ARP}} &= L \times (M-1) \times 60 \times 8/T_D \\ &\quad + (M-1) \times L \times 84 \times 8/T_D \\ &\quad + (M-1) \times (M-2) \times 60 \times 8/T_D \\ &= \frac{\{1.15L + 0.48(M-2)\}(M-1)}{T_D}\text{Kbps,} \end{aligned} \tag{6}$$

where the first term on the right hand side of the first equality describes the rate of the voting reply packets triggered by Node A itself, the second term describes the traffic rate of voting request packets received by Node A, and the third term is the rate of the voting reply packets triggered by the special voting request messages of other EMR-ARP nodes.

By comparing (5) and (6), we can find that the voting traffic overhead of EMR-ARP is lower than that of MR-ARP if $L \geq 2$. If $L \geq 10$, then the voting traffic overhead of EMR-ARP is lower than that of MR-ARP by a factor of more than six since the EMR-ARP does not require multiple transmission of the same voting reply packets. When $L = M = 255$, and T_D is one day, $r_A^{\text{EMR - ARP}} = 1.22$Kbps.. Thus, the voting traffic overhead is not significant for a small subnet.

4.2 Analysis of reliability of the proposed scheme

Before investigating the reliability of the proposed scheme in the presence of attackers in detail, we first investigate whether the fairness can be maintained even when the performance of the CPUs is not the same among different machines.

We implemented the proposed EMR-ARP mechanism on a 2.66 GHz quad-core PC by modifying the ARP-related code of Fedora 9 Linux (kernel 2.6.25). As explained in Section 3.1, each puzzle is solved by applying the message transformation function F_m to a given message, and the puzzle solution is verified by applying the message recovery function G_m to a given transformed message. Among the message transformation function-related parameters, e is fixed to 3 as explained already. The values of N and d are selected once, according to the rule described in Section 3.1 under the constraint that N and d are 256 bits long, and the selected values of N, d, and e are stored in the ARP kernel code, so that they can be shared between different machines by just installing the same or consistent version of OS. Figure 7 compares the message transformation time and the message recovery time, i.e., the puzzle solving time and the puzzle solution verification time, for various values of iteration number m. Since the puzzle solving time increases linearly with respect to m, we can control the complexity of the puzzle by m. We can also observe that it takes a much longer time to solve a puzzle than to verify the given puzzle solution. In Figure 7, the message transform time is larger than the message recovery time by a factor of about 20. This ratio can be increased further if we increase the bit length of N, since the bit length of d should increase accordingly, while e is fixed to 3. However, if the size of N increases, then the sizes of field c_s in the voting request message and field c_r in the voting reply message should also increase, yielding higher voting traffic overhead. We fix the size of N to 256 bits to keep the voting traffic overhead small.

We also determine the value of m based on the result shown in Figure 7. The message transformation time, i.e., puzzle solving time, should be sufficiently longer than the MAC layer access delay of 802.11 wireless LAN so that the different transmission rate on the wired and wireless medium cannot impair the fairness in voting among the wired and wireless nodes [22]. Thus, we select a sufficiently large value, i.e., 4000, for m to maintain the message transformation time over 450 ms. However, since the optimal value of m can

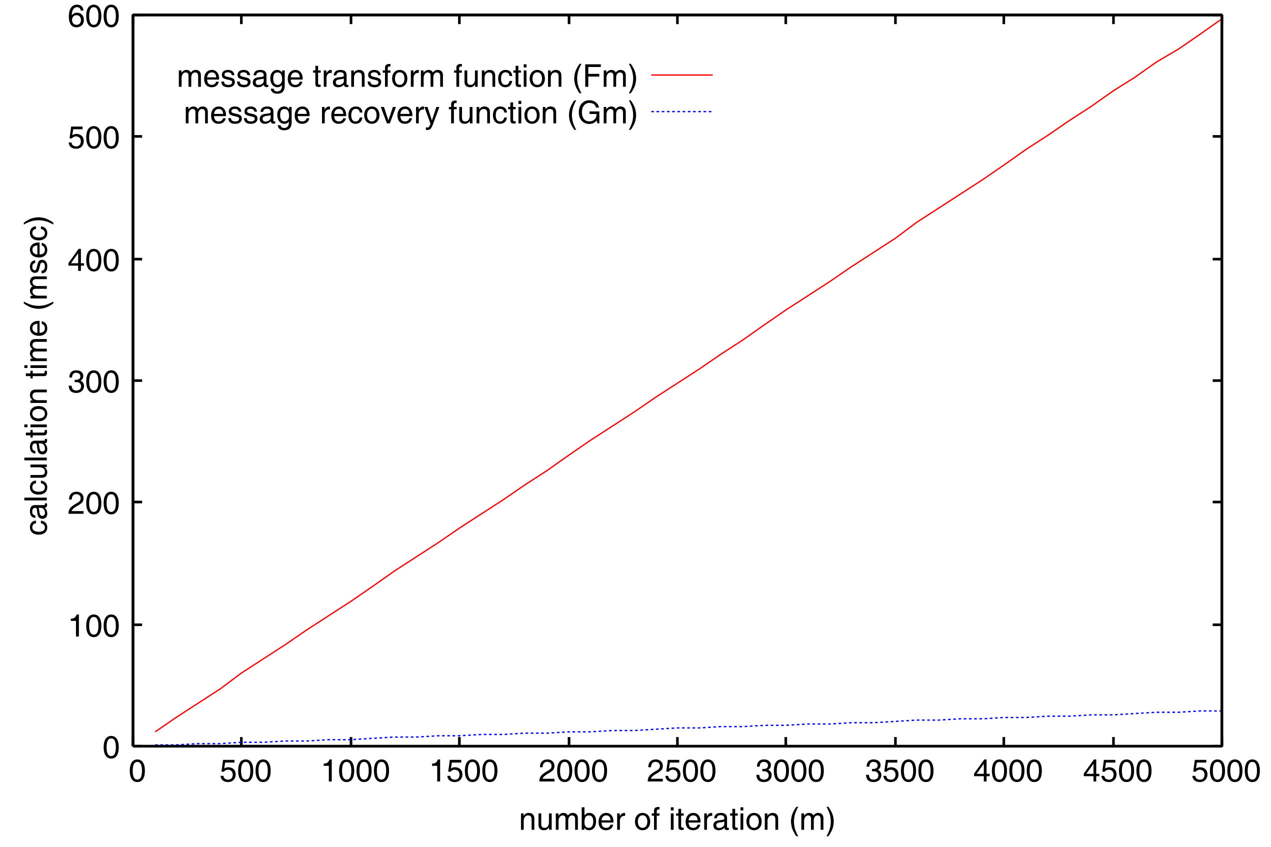

Figure 7 Comparison of message transformation time (by $F_m(x)$) and message recovery time (by $G_m(x)$) for various values of m.

differ depending on the machine, the value of m can be determined according to the above rule by each machine whenever it is rebooted.

Next, we investigate whether the fairness in voting can be maintained even when multiple machines with different CPUs are involved in the same voting process. Figure 8 compares the message transformation times obtained from the three machines with different CPUs, i.e., 3 GHz Pentium D, 2.33 GHz Core2duo, and 2.66 GHz Core2quad processors, respectively. The quad-core processor yields the shortest computation time, and the Pentium processor gives the longest computation time. The dotted line in Figure 8 represents an upper bound corresponding to two times the average message transformation time of the quad-core machine. We can observe that the ratio of the maximum computation time to the minimum computation time is less than 2. The standard deviation of the computation time was measured to be less than 1% of the mean value in all the cases.

As an example, let us consider a case where the processors of all the machines belonging to the given subnet are one of the above three types of CPUs. Let us assume that the puzzle solving times of different machines do not differ by a factor of two, as shown in Figure 8. Then, even though an attacker uses the quad-core processor and tries to send multiple voting replies after solving many puzzles for other MAC addresses, so as to pretend to be multiple other machines, the first voting replies from other normal EMR-ARP nodes will arrive earlier than the second voting reply of the malicious node if all the nodes experience a similar level of MAC layer access delay. If t_1 denotes the response time of the earliest voting reply packet, then the voting request node does not accept the voting reply packets that do not arrive within $2t_1$ from the voting request transmission time as explained in Section 3.1. Thus, the fairness can be guaranteed if the puzzle solving times of different machines do not differ by a factor of two. Figure 8 implies that this condition is likely to be valid in a real environment since the performance of the CPUs used in recent desktop and laptop computers is usually not much different from that of the three types of CPUs considered in this article.

We now investigate how well the proposed EMR-ARP scheme resists the ARP cache spoofing attack when

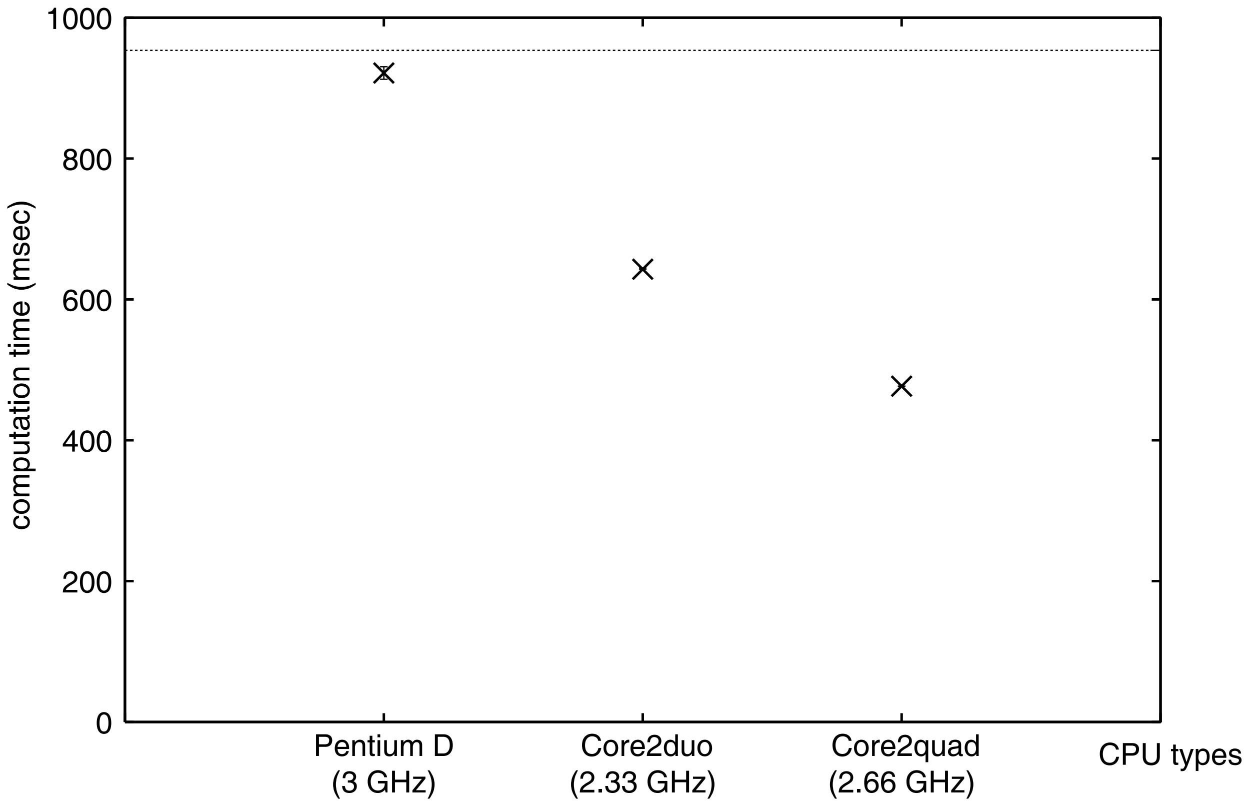

Figure 8 Comparison of message transformation times, i.e., puzzle solving times, among the machines with different types of CPUs (*m* = 4000).

attack nodes exist. In more detail, we compare the proposed EMR-ARP scheme with MR-ARP [13] in terms of the false decision probability in the presence of attackers. Since we assume that all the malicious nodes collude to raise the ratio of the votes supporting a selected fake MAC address, false decision is made when the aggregate number of votes from the adversaries exceeds 50% of the total valid votes. Figure 9 shows the network topology for the test-bed to evaluate both MR-ARP and EMR-ARP. There are seven MR-ARP or EMR-ARP nodes, denoted *Ni* (*i* = 1,2,...,7). Among them, five nodes from *N*1 to *N*5 are connected to the subnet through 802.11g wireless link. The remaining two nodes *N*6 and *N*7 are connected to the LAN through Ethernet switch. Although we use simple a linux machine for *N*6, we assume that *N*6 plays the role of a gateway router, and thus, each newly attached node tries to resolve the MAC address of *N*6 first. In our experiment scenario, *N*7 is newly attached. We focus on the voting procedure that is triggered by *N*7 to resolve the MAC address of *N*6. Although five adversary nodes, *Ai* (*i* = 1, 2,...,5), are shown in Figure 9, the number of attackers is changed from one to five depending on the experiment scenarios. We connected the adversary nodes to the LAN through a 100 Mbps ethernet switch to consider the situation where the attackers are in a more advantageous position than the normal MR-ARP or EMR-ARP nodes in terms of the transmission speed. In addition, we used the machines with a better CPU performance for attackers to test the worst case under the given condition. In more detail, we used five quad-core machines for attackers, one pentium-D machine for the gateway router, one quad-core machine for the new node, i.e., voting request node, and three dual-core and two quad-core machines for other MR-ARP or EMR-ARP nodes.

Figure 10 compares the false decision probability obtained by EMR-ARP with that obtained by MR-ARP for various numbers of adversary nodes under the condition described above. We performed 50 experiments for each number of adversary nodes. We can observe that the MR-ARP suffers from very high false decision probability, even when the number of adversary node is only one, i.e., lower than the total number of MR-ARP nodes participating in the voting, six. The reason can be explained as follows. There are five wireless MR-ARP nodes. However, the voting reply packets from those wireless machines always arrive very late compared to the ones from wired nodes due to the MAC-layer access

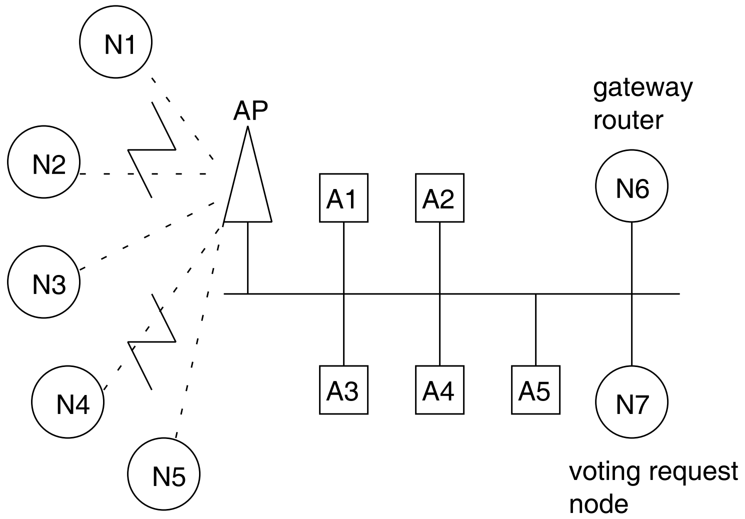

Figure 9 Test-bed network topology to evaluate MR-ARP and the proposed EMR-ARP.

delay of at least tens of milliseconds. Although there are one wired MR-ARP node and one wired adversary node, the attacker can send more than 50 reply packets while the MR-ARP node replies with only 50 reply messages. Thus, a single attacker can always win the voting in case of MR-ARP. In contrast, the false decision probability is maintained very low in case of EMR-ARP since the sufficiently long puzzle computation time resolves the unfairness issue caused by unbalanced transmission rates or MAC-layer access delay in wired and wireless network. Thus, EMR-ARP can effectively protect the upgraded nodes, as long as the number of the EMR-ARP nodes is larger than that of the adversary nodes, when the CPU performance of the normal EMR-ARP machines and the attacker machines does not significantly differ.

We briefly discuss the compatibility of EMR-ARP with the existing ARP. EMR-ARP Node A responds to a normal ARP request destined to itself by sending an ARP reply message as the current ARP. Even though Node A is the only EMR-ARP-enabled node in the subnet, Node A can accept the received IP/MAC mapping for a new IP address using the rule for (C) in Figure 1. Thus, EMR-ARP is backward compatible with the existing ARP.

4.3 Analysis of conflict resolution delay

In this section, we investigate how long it takes to resolve the conflict on IP/MAC address mapping especially in the cases (B) and (C) of Figure 1. In case (B) of Figure 1, the conflict is resolved by sending 10 ARP request packets to the previous owner of the IP address in question and receiving an ARP response packet from that node. Since the puzzle computation is not involved in this step and the packets are hardly dropped in the normal situation without any DoS attack, it usually takes only one round-trip time (RTT) between the querying node and the previous owner node to resolve the conflict in this case. Since the querying node waits up to 1 s, the conflict resolution time does not exceed 1 s even though there is no response from the previous owner node.

In case (C) of Figure 1, the voting request node needs to perform the following tasks to make a decision on the owner of the IP address in question. The voting request node first generates a first-type puzzle, solves it, generates a key for the second-type puzzle, broadcasts a voting request packet containing the solution for the first-type puzzle and the key for the second-type puzzle, waits for the voting reply messages from the neighbor nodes only up to 1 s as described in Section 3.1, verifies the solutions for the second-type puzzle, and makes a decision based on the votes from the nodes providing the correct solutions. Thus, the total conflict resolution time W can be expressed as

$$W = W_{p1\_\text{gen}} + W_{p1\_\text{sol}} + W_{p2\_\text{gen}} + W_{\text{wait}} + W_{v} + W_{d},$$

where W_{p1_gen} is the first-type puzzle generation time, W_{p1_sol} is the duration of the time interval to solve the first-type puzzle, W_{p2_gen} is the second-type puzzle generation time, W_{wait} is the waiting time for the voting reply messages, i.e., 1 s, W_v is the time to verify the solutions for the second-type puzzles, and W_d is the time to make a decision based on the valid votes. As described in Section 3.1, the first-type puzzle is generated by simply concatenating its own MAC address and the local time, and the key for the second-type puzzle is generated through a combination of concatenating and

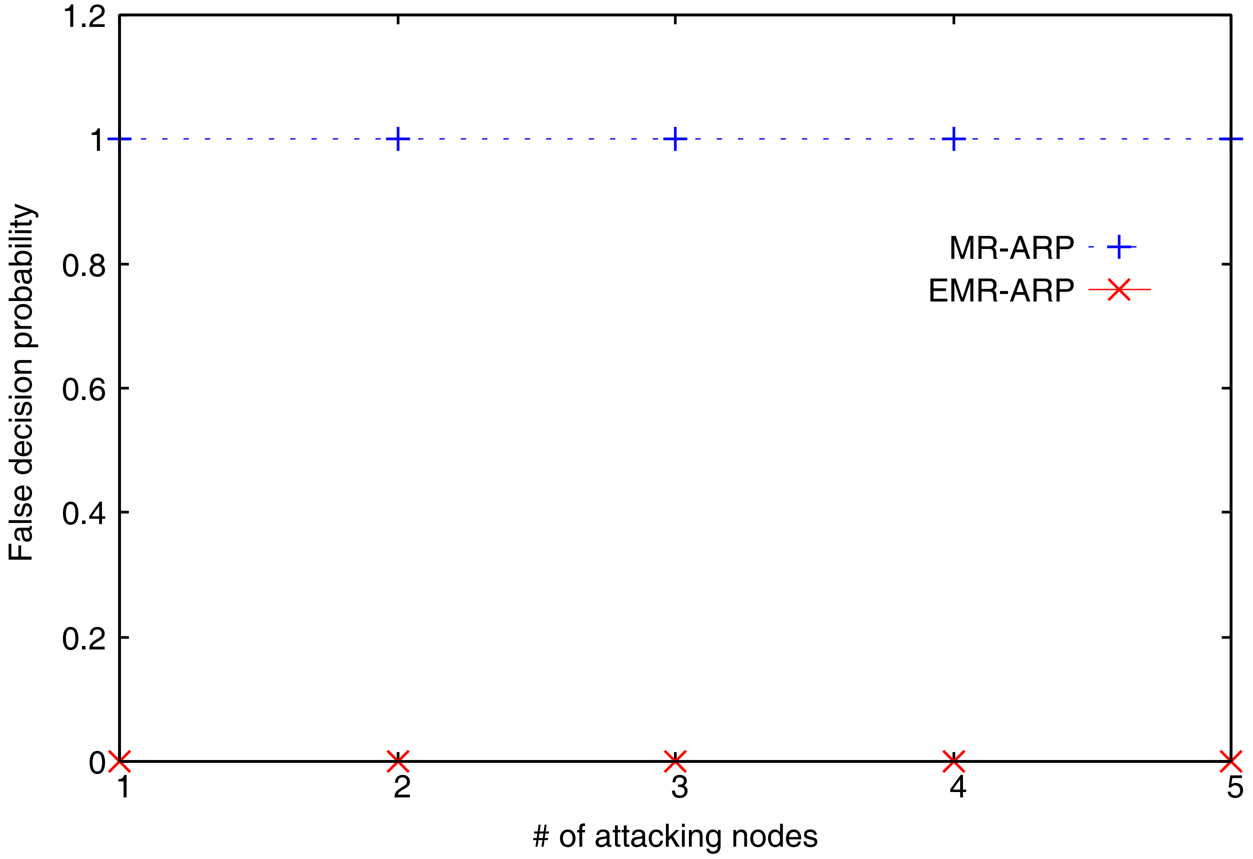

Figure 10 Comparison of false decision probabilities of EMR-ARP and MR-ARP for various numbers of adversary nodes when the number of EMR-ARP/MR-ARP nodes is 6.

a single hash function operation as described in (4). Thus, the three components, W_{p1_gen}, W_{p2_gen}, and W_d, are negligibly small compared with other components, and the above expression for W can be simplified to

$$W \simeq W_{p1\_sol} + 1\ \mathrm{s} + W_v.$$

W_{p1_sol} is usually around 450 ms since the value of the puzzle-related parameter m is selected to maintain W_{p1_sol} close to 450 ms as explained in the previous section. The puzzle solution verification time is much shorter than the puzzle solving time as explained in the section. Thus, W_v is usually smaller than W_{p1_sol}, but W_v increases linearly with the number of received voting reply messages. We performed experiment to investigate the trend of W as the number of EMR-ARP nodes increases. Figure 11 shows the range of the voting procedure delay (W), i.e., $[\mu - \sigma, \mu + \sigma]$, where μ and σ are the mean and the standard deviation of W obtained after 20 experiments for each number of neighbor EMR-ARP nodes. We can observe that the voting procedure delay W increases slowly from 1.5 s as the number of the neighbor EMR-ARP nodes increases. The increasing trend is due to W_v, that is proportional to the number of the received voting reply messages. However, since the voting request node waits for the voting reply messages only up to 1 s from the transmission time of the voting request packet, the number of voting reply packets received during a 1 s interval is likely to be limited due to highly variable MAC layer access delay and diverse computation power of the neighbor nodes in a real environment, and the correct decision can be made as long as the votes from the normal EMR-ARP nodes outnumber the valid votes from the adversary nodes. Thus, W is highly likely to saturate to a similar level even though the number of EMR-ARP nodes increases further.

5 Conclusions and future work

An enhanced version ARP, EMR-ARP, is proposed to prevent ARP cache poisoning-based MITM attacks in wired or wireless LAN environments. The proposed scheme is based on two key concepts: long-term IP/MAC mapping table and computational puzzle-based voting. The long-term table protects the IP/MAC

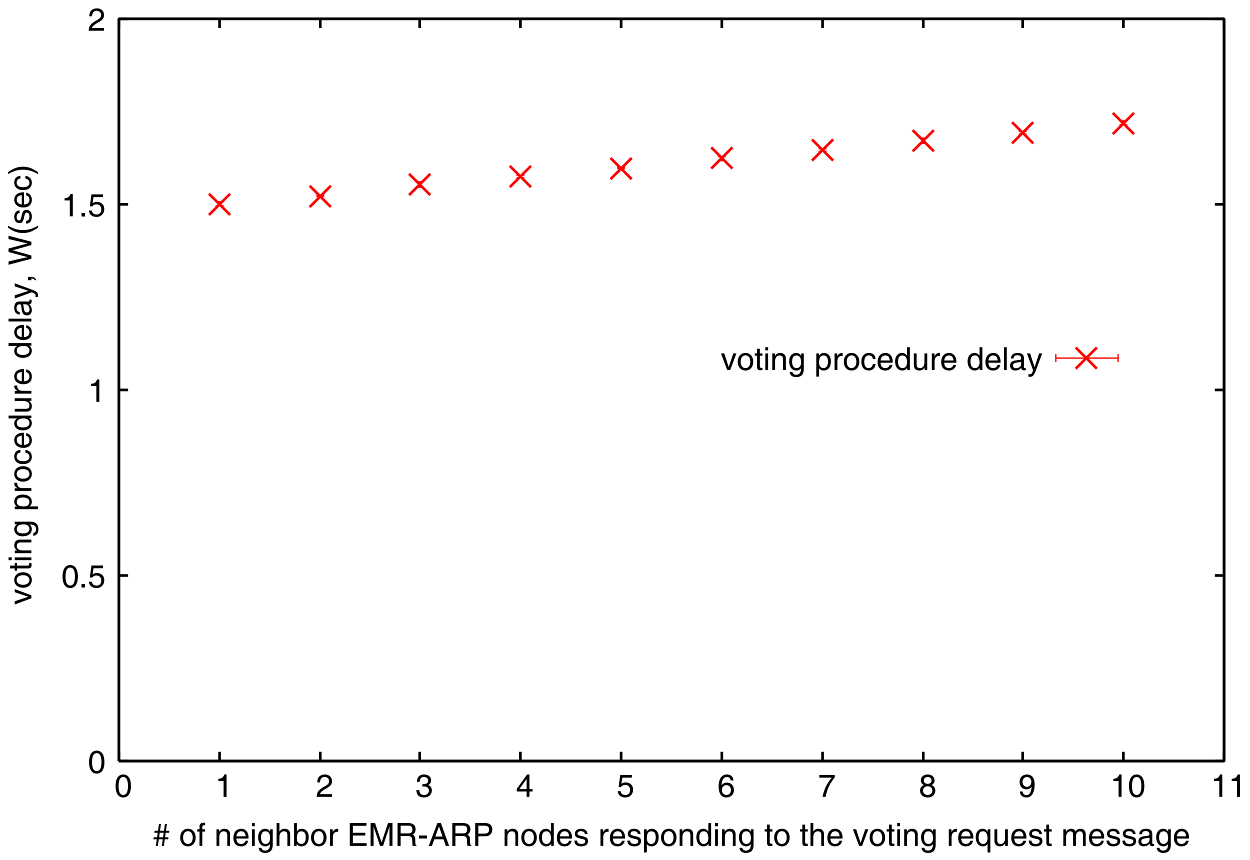

Figure 11 Voting procedure delay for various numbers of neighbor EMR-ARP nodes.

address mappings for all alive machines in the subnet from the ARP cache poisoning attack. The computational puzzle-based voting prevents ARP poisoning-based MITM attack whenever a machine is rebooted or a new machine is added. It is very important to maintain the fairness between different machines when the MAC addresses are resolved based on voting. We designed a new puzzle based on the RSA algorithm to provide fairness, even when the link rates differ between wireless and wired nodes, while minimizing the variance in puzzle solving time.

The experiment results show that the ARP poisoning-based MITM attacks are well mitigated, as long as the number of EMR-ARP nodes is larger than that of adversary nodes. Since the proposed mechanism is not based on public-key cryptography, the manual configuration, such as distribution of the public key and the MAC address of the centralized key management server, is not required, and thus, the proposed scheme can efficiently mitigate ARP poisoning-based MITM attacks, even in public Wi-Fi hot-spots. The proposed scheme is backward compatible with existing ARP protocol and incrementally deployable, with benefits to the upgraded machines.

The proposed computational puzzle-based voting scheme can provide fairness among different nodes only when the computation power of the CPUs of those machines is not significantly different. The proposed scheme will be extended further, so that the fairness can be maintained, even when the computation power of the machines is more diverse in future study.

Acknowledgements
This research was supported by the Yeungnam University research grants in 2010.

Competing interests
The authors declare that they have no competing interests.

Received: 31 May 2011 Accepted: 6 March 2012
Published: 6 March 2012

References
1. DC Plummer, An ethernet address resolution protocol. RFC. **826** (1982)
2. RW Stevens, *TCP/IP Illustrated*, vol. 1. (Addison Wesley, Boston, 2001)
3. C Benvenuti, *Understanding Linux Network Internals*, (O'Reilly, California, 2006)

4. Hacking UNIX, A tutorial for performing various attacks including ARP poisoning attack on UNIX systems. (2003) http://duho.cjb.net
5. S Whalen, An introduction to arp spoofing. http://packetstormsecurity.org/papers/protocols/intro_to_arp_spoofing.pdf
6. V Goyal, R Tripathy, An efficient solution to the ARP cache poisoning problem, in *Proc of Australasian Conference on Information Security and Privacy (ACISP)*, vol. 1. Brisbane, Australia, pp. 40–51 (July 2005)
7. B Fleck, J Dimov, Wireless access points and arp poisoning. http://bandwidthco.com/whitepapers/netforensics/arp-rarp/Wireless%20Access%20Points%20and%20ARP%20Poisoning.pdf
8. Y Bhaiji, *Network Security Technologies and Solutions*, (Cisco Press, New York, 2008)
9. D Bruschi, A Ornaghi, E Rosti, S-ARP: a Secure address resolution protocol, in *Proc of Annual Computer Security Applications Conference (ACSAC)*, vol. 1. Las Vegas, Nevada, USA, pp. 66–74 (Dec. 2003)
10. W Lootah, W Enck, P McDaniel, TARP: Ticket-based address resolution protocol. Comput Netw. **51**(15), 4322–4337 (2007). doi:10.1016/j.comnet.2007.05.007
11. I Teterin, Antidote. http://online.securityfocus.com/archive/1/299929
12. R Philip, Securing Wireless Networks from ARP Cache Poisoning, Master's Thesis, San Jose State University, (2007)
13. SY Nam, D Kim, J Kim, Enhanced ARP: preventing ARP poisoning-based man-in-the-middle attacks. IEEE Commun Lett. **14**(2), 187–189 (2010)
14. A Kamerman, L Monteban, WaveLAN-II, A high-performance wireless LAN for the unlicensed band. Bell Lab Techn J. **2**(3), 118–133 (1997)
15. M Lacage, MH Manshaei, T Turletti, IEEE 802.11 rate adaptation: a practical approach, in *Proc of ACM International Symposium on Modeling, analysis and simulation of wireless and mobile systems (MSWiM'04)*, vol. 1. Venice, Italy, pp. 126–134 (Oct 2004)
16. C Dwork, M Naor, Pricing via processing or combatting junk mail, in *Proc of CRYPTO*, vol. 1. Santa Barbara, California, USA, pp. 139–147 (Aug 1992)
17. N Borisov, Computational puzzles as sybil defenses, in *Proc of IEEE International Conference on Peer-to-Peer Computing*, vol. 1. Cambridge, UK, pp. 171–176 (Sep 2006)
18. B Parno, D Wendlandt, E Shi, A Perrig, B Maggs, YC Hu, Portcullis: protecting connection setup from denial-of-capability attacks, in *Proc of SIGCOMM*, vol. 1. Kyoto, Japan, pp. 289–300 (Aug 2007)
19. RL Rivest, A Shamir, L Adleman, A method for obtaining digital signatures and public-key cryptosystems. Commun ACM. **21**(2), 120–126 (1978). doi:10.1145/359340.359342
20. C Kaufman, R Perlman, M Speciner, *Network Security-Private Communication in a Public World*, 2nd edn. (Prentice Hall, Upper Saddle River, 2002)
21. Gratuitous ARP - The Wireshark Wiki, http://wiki.wireshark.org/Gratuitous_ARP
22. P Chatzimisios, AC Boucouvalas, V Vitsas, IEEE 802.11 packet delay-a finite retry limit analysis, in *Proc of IEEE Globecom*, vol. 2. San Francisco, USA, pp. 950–954 (Dec 2003)

doi:10.1186/1687-1499-2012-89
Cite this article as: Nam *et al.*: **Mitigating ARP poisoning-based man-in-the-middle attacks in wired or wireless LAN.** *EURASIP Journal on Wireless Communications and Networking* 2012 **2012**:89.