An Efficient Solution to the ARP Cache Poisoning Problem

Vipul Goyal and Rohit Tripathy

OSP Global, Town Center, Andheri(E),
Mumbai, India
{vgoyal, rtripathy}@ospglobal.com

Abstract. ARP cache poisoning is a long standing problem which is known to be difficult to solve without compromising efficiency. The cause of this problem is the absence of authentication of the mapping between IP addresses and MAC addresses. Due to lack of the required authentication, any host on the LAN can forge an ARP reply containing malicious IP to MAC address mapping causing ARP cache poisoning. In fact, there are a number of tools freely available on the internet using which, even a newbie can launch such an attack. In this paper, we present a new cryptographic technique to make ARP secure and provide protection against ARP cache poisoning. Our technique is based on the combination of digital signatures and one time passwords based on hash chains. This hybrid system prevents the ARP cache poisoning attack while maintaining a good system performance at the same time.

1 Introduction

Local Area Networks running TCP/IP over Ethernet are the most common networks these days. Each host on such a network is assigned an IP address (32 bits). Hosts also posses a network interface card (NIC) having a unique physical address (48 bits) also called the MAC address. For the final delivery of any packet destined to some host, its MAC must be known to the sender. Thus, the address resolution protocol is used to resolve an IP address into a MAC address. Resolved addresses are kept in a cache so as to avoid unnecessary work for already resolved addresses every time they are needed. Resolution is invoked only for entries expired or absent from the cache, otherwise cache entries are used.

The ARP Poisoning attack involves maliciously modifying the association between an IP address and its corresponding MAC address so as to receive the data intended to someone else (victim). By performing ARP poisoning, an attacker forces a host to send packets to a MAC address different from the intended destination, which may allow her to eavesdrop on the communication, modify its content (e.g., filtering it, injecting commands or malicious code) or hijack the connection. Furthermore, when performed on two different hosts at the same time, ARP poisoning enables an adversary to launch a "man in the middle" (MiM) attack. With MiM attacks, the traffic between two hosts is redirected through a third one, which acts as the man in the middle, without the two knowing it. The MiM may simply relay the traffic after inspecting it or modify it before resending it. Note that MiM attacks are also possible at various other network layers. ARP cache poisoning allows performing such an attack at data link layer.

ARP Cache poisoning can also be used to launch a Denial-of-Service (DoS) attack [20]. Furthermore, this attack is not just confined to Ethernet networks but layer 2 switched LANs and 802.11b networks are also vulnerable. In [2], various scenarios are described where a wireless attacker poisons two wired victims, a wireless victim and a wired one, or two wireless victims, either through different access points or a single one. Even a newbie can launch sophisticated poisoning attack using easily available tools and tutorials on internet, [5, 7, and 10].

In this paper, we propose a solution to the ARP cache poisoning problem based on an extension of the ARP protocol. We introduce a set of functionalities that enable an integrity and authenticity check on the content of ARP replies, using a combination of digital signatures and one time passwords based on hash chains.

Rest of the paper is organized as follows. Section 2 illustrates the problem considered in this paper and recalls how ARP works and why it is vulnerable to cache poisoning. Section 3 discusses the related work. Section 4 describes the proposed solution. Section 5 concludes the paper.

2 Problem Definition

2.1 Address Resolution Protocol

Hosts and applications in a network work with domain names which are converted to the IP address by a DNS server. But once packets containing application data arrive on the local Ethernet network of the host, they can be transmitted only if the MAC address buried in the NIC of the destination host is known to the switch. Thus a conversion is needed from IP addresses to MAC addresses and vice versa. This conversion is done by the Address Resolution Protocol or ARP in short [8, 6].

ARP works as follows. When a host needs to send an IP datagram as an Ethernet frame to another host whose MAC address it does not know, it broadcasts a request for the MAC address associated with the IP address of the destination. Every host on the subnet receives the request and checks if the IP address in the request is bound to one of its network interfaces. If this is the case, the host with the matching IP address sends a unicast reply to the sender of the request with the <IP address, MAC address> pair. Every host maintains a table of <IP, MAC> pairs, called the ARP cache, based on the replies it received, in order to minimize the number of requests sent on the network. No request is made if the <IP, MAC> pair of interest is already present in the cache. ARP cache entries have a typical lifetime of 20 minutes, after which the entry should be refreshed.

In ARP, a reply may be processed even though the corresponding request was never received, i.e., it is a stateless protocol. When a host receives a reply, it updates the corresponding entry in the cache with the <IP, MAC> pair in the reply. While a cache entry should be updated only if the mapping is already present, some operating systems, e.g., Linux and Windows, cache a reply in any case to optimize performance.

2.2 ARP Cache Poisoning

Since ARP replies are not authenticated, an attacker can send an ARP reply containing a malicious <IP, MAC> association to any host on the network thus poisoning the ARP cache of that host. The attacker may supply her MAC address in the sent malicious association which enables her to receive all the packets sent by that host to the IP address specified in the association. This way the attacker may receive all the frames originally directed to some other host. If also the cache of the real destination host is poisoned, both communication flows are under the attacker's control. In this situation, the attacker could launch a man in the middle, where she can forward the received packets to the correct destination after inspecting and possibly modifying them. The two end points of the communication will not notice the extra hop added by the attacker if the packet TTL is not decremented.

Although cache can easily be poisoned when there is an entry in the cache for the targeted IP address, some operating systems, e.g. Solaris, will not update an entry in the cache if such an entry is not already present when an unsolicited ARP reply is received. Although this might seem a somewhat effective precaution against cache poisoning, the attack is still possible. The attacker needs to trick the victim into adding a new entry in the cache first, so that a future (unsolicited) ARP reply can update it. By sending a forged ICMP echo request as if it was from one of the two victims, the attacker has the other victim create a new entry in the cache. When the other victim receives the spoofed ICMP echo request, it replies with an ICMP echo reply, which requires resolving first the IP address of the original ICMP request into an Ethernet address, thus creating an entry in the cache. The attacker can now update it with an unsolicited ARP reply.

3 Related Works

ARP-based attacks are not easily prevented in current architectures. There are a handful of actions often recommended for mitigation. The first of these is employing static ARP, which renders entries in an ARP cache immutable. Thus any address resolution protocol is not employed at all. This is currently the only true defense [17], but is impractical. Windows machines ignore the static flag and always update the cache. In addition, handling static entries for each client in a network is unfeasible for all but the smallest networks. An administrator must deploy new entries to every machine on the network when a new client is connected, or when a network interface card (NIC) is replaced. Furthermore, this prevents the use of some DHCP configurations which frequently change MAC/IP associations during lease renewal.

The second recommended action is enabling port security on the switch. Also known as MAC binding, this is a feature of high-end switches which ties a physical port to a MAC address. This fixed address can be manually set by the administrator to a range of one or more addresses, or can be auto-configured by the switch during the first frame transmission on the port. These port/address associations are stored in Content Addressable Memory (CAM) tables [15], a hardware-based reverse lookup device. A change in the transmitter's MAC address can result in port shutdown, or other actions as configured by the administrator. However, port security is far from ubiquitous and does nothing to prevent ARP spoofing [16]. Consider a man-in-the-middle attack as presented in [11]. An attacker X only needs to convince victim A to deliver frames meant for B to X, and vice versa for victim B. When sending forged ARP replies to achieve this, at no time must X forge its MAC address – only the cache of the clients is manipulated. Port security validates the source MAC in the frame header, but ARP frames contain an additional source MAC field in the data payload, and it is this field that clients use to populate their caches [6]. It should be said, however, that port security does prevent other attacks mentioned in [11] such as MAC flooding and cloning.

Virtual LANs (VLANs) create network boundaries which ARP traffic cannot cross, limiting the number of clients susceptible to attack. However, VLANs are not always an option and have their own set of vulnerabilities as detailed in [18].

Arpwatch [11] allows notification of MAC/IP changes via email. IDS and personal warn the user that the entry in the cache is changed. In these solutions, the decision is left to the user and his/her awareness. Given the particularly sophisticated level of operation in this case, it is doubtful that the average user will take the proper actions. Still, detection is an important step in mitigation. Solutions such as a centralized ARP cache or a DHCP server broadcasting ARP information, as they are deployed in IP over ATM networks [4], have their own problems as the attacker could spoof the source of the broadcast and poison the whole LAN [12].

Some kernel patches exist that try to defend against ARP poisoning. "Anticap" [1] does not update the ARP cache when an ARP reply carries a different MAC address for a given IP from then one already in cache and will issue a kernel alert that someone is trying to poison the ARP cache. "Antidote" [9] is more sophisticated. When a new ARP replies announcing a change in a <IP, MAC> pair is received, it tries to discover if the previous MAC address is still alive. If the previous MAC address replies to the request, the update is rejected and the new MAC address is added to a list of "banned" addresses. If Antidote is installed, an attacker may spoof the sender MAC address and force a host to ban another host.

The only kernel patch which assures mutual authentication between the requester and the replier even on the first message is Secure Link Layer [3]. SLL provides authenticated and encrypted communication between any two hosts on the same LAN. SLL requires a Certification Authority (CA) to generate SLL certificates for all legitimate hosts on the network. SLL handles authentication and session key exchange before any messages are transferred from one host to another. Elliptic curve cryptography algorithms are used for both operations. SLL defines three authentication messages that hosts send each other to perform mutual authentication and session key exchange. After authentication, the payload data field of all Ethernet frames sent between two hosts is encrypted with Rijndael using a 128-bit key and 128-bit long blocks. However, such a mechanism is too slow and complex for the purpose of ARP. Mutual authentication between two hosts is sufficient for avoiding ARP poisoning. Encrypting ARP replies does not yield any additional security since the association between IP and MAC addresses should be public. Furthermore, SLL also maintains all the cryptographic keys in kernel-space. Note that the amount of memory required could be considerable in case of class B networks. It is not recommended to use kernel memory with information that could be as well managed in user space, such as keys. Hence although SLL is sufficiently secure, it has an unacceptable impact on the system performance.

S-ARP was recently proposed [12]. Each host has a public/private key pair certified by a local trusted party on the LAN, which acts as a Certification Authority. Each ARP reply is digitally signed by the sender, thus preventing the injection of malicious replies. At the receiving end, the cache entry is updated if and only if the signature is correctly verified. Cryptographic keys are maintained in the user space. Unnecessary services provided by SLL and not required for the sake of ARP are removed. Thus S-ARP is more efficient than SLL.

SLL and S-ARP are probably the only secure solutions for preventing the ARP cache poisoning. We however note that S-ARP still requires all ARP replies to be digitally signed. Recall that asymmetric key cryptography and digital signatures are considerably slow when compared to symmetric key cryptography and one way hash functions. Roughly, RSA signature generation is about 10,000 times slower than calculation of a one way hash function [19]. Thus, even S-ARP may have unacceptable impacts on the system performance.

4 The Proposed Solution

4.1 The Basic Idea

Our solution for the prevention of ARP cache poisoning is based on a combination of digital signatures and one time passwords. One time passwords are based on hash chains. Cryptographic techniques can hardly be avoided as the receiver has to authenticate the ARP reply; however an intelligent use of cryptography is desired to avoid unacceptable performance penalties.

Our protocol requires periodic generation of digital signatures, the rate of generation being independent of the number of ARP requests being received. For this, we identify two different components of an ARP reply:

1. The <IP address, MAC address> mapping
2. The recency of the mapping

The first component requires a digital signature since the <IP, MAC> mapping must be authentic and its authenticity must be publicly verifiable. Our idea is to however to use the same digital signature again and again in ARP replies for a long time.

Here one option could be to trust the ARP reply for recency and the only check performed on the content of replies would be validation of the digital signature. But then an attacker could get hold of that digital signature by simply sending an ARP request to the target system and getting it in reply. It could then wait for the target system to go down or it could crash the target system using known attacks or Denial of Service attacks. As soon as the target system goes down or gets disconnected from the network, the attacker could change her MAC address to that of the target system and thus receive all packets sent to it. Now, even when the target system comes up later, it cannot claim back its MAC address and has to change it. The attacker may continue to poison the ARP cache of other hosts using the stored digital signature and thus receive the packet sent to the target system.

Hence we need a method to somehow securely indicate the recency of the mapping indicated in the digital signature. This is done by including a one time password in the ARP reply. Thus, the basic idea of our protocol is:

Generate a digital signature *S* containing the IP address to MAC address mapping, the local clock time and the tip of a hash chain used for verifying one time passwords. Now, for the first 20 minutes (cache entry validity time), the host answers ARP requests by sending *S* as the ARP reply. For the next 20 minutes, the host sends *S* and the first one time password (first link of the hash chain) as ARP reply and so-on. In general for the i^{th} 20 minute slot, the host sends *S* and the $(i\text{-}1)^{th}$ one time password as the ARP reply.

This process is continued till the one time passwords (or the links of the hash chain) do not get exhausted or the <IP, MAC> association of the host does not change. After this, a new signature S' should be generated and the whole process be repeated.

We now describe the one time password system being used and then move on the detailed description of the proposed protocol using that.

4.2 One Time Passwords

We use a variant of the one time password system designed by Leslie Lamport [13, 14]. This system is popularly known as Lamport Hashes or S/KEY. This scheme is used to authenticate a client to an untrusted server and is based upon the concept of Hash chains. No security sensitive quantities are stored at the server, i.e., the password verifying token is public.

The one time passwords are generated using a secret *K* known only to the client. The client chooses an integer *N* and a random number *R* acting as the nonce. For system initialization, the client then somehow securely sends *N* and $H^{N+1}(K||R)$ to the server[1] (e.g. using digital signatures).

At any point of time, the server maintains the following 3-tuple entry for each client:

$$\boldsymbol{< id, n, H^{n+1}(K||R) >} \quad \text{with } n=N \text{ initially}$$

The client authenticates by sending $H^n(K||R)$ to the server (along with its id). The server computes its hash and then compares it with the stored $H^{n+1}(K||R)$. If they match, $H^{n+1}(K||R)$ is replaced with $H^n(K||R)$, the value of stored *n* is decremented and the client is successfully authenticated.

When *n* reaches 0 at the server, i.e., when the client authenticates with $H(K||R)$, the list of one time passwords is considered to be exhausted. At this point, a new value of *R* must be chosen and the system should be reinitialized.

[1] H is a one way hash function like MD5 and || denotes concatenation

4.3 Network Setup

The setup phase in our system is similar to that in S-ARP. Every host on the network is identified by its own IP address and has a public/private key pair. Besides, there is a trusted host on the network called the Authoritative Key Distributor (AKD) which handles the task of key distribution and clock synchronization.

Note that this AKD based architecture can easily be converted to Certificate Authority (CA) based architecture. This can be done by distributing a certificate containing the IP address to public key mapping to each host on the network. We elaborate more on this issue in section 4.6.1.

The first step when setting up a LAN that uses our protocol is to identify the AKD and distribute through a secure channel its public key and MAC address to all the other hosts. Such an operation may be performed manually when a host is installed on the LAN for the first time. On the other hand, a host that wants to connect to the LAN must first generate a public/private key pair and send the public key along with its IP address to the AKD. Here the correctness of the information provided is verified by the network manager and the host public key together with its IP address is entered in the AKD repository. This operation has to be performed only the first time a host enters the LAN. If a host wants to change its key, it communicates the new key to the AKD by signing the request with the old one. The AKD will update its key and the association is correctly maintained. Section 4.6 explains the protocol behavior when IP addresses are dynamically assigned by a DHCP server.

Once connected to the LAN, a host synchronizes its local clock with the one received from the AKD. To avoid the reply of old clock value from an adversary during clock synchronization, the host generates a random number R which it sends along with the synchronization request to the AKD. The AKD replies back with the current time t along with a digital signature on (t, R).

4.4 Message Format

The ARP request message format remains the same except for the addition of two new fields- “timestamp” and “type”. “timestamp” is the value of the local clock at the time of request generation. The value of “type” field may either be 1 or 2 to distinguish among the following two types of requests:

1. New entry request
2. Entry refreshment request

We discuss in the next section about how a host determines the type of request to send in any scenario.

Our protocol adds an extension to the ARP reply header. The extension comprises of a new field called “type” and a variable length payload called “data”. The field “type” distinguishes among the following six types of messages:

1. New entry creation (reply only)
2. Entry refreshment (reply only)
3. Public key management (request/reply)
4. Time synchronization (request/reply)

Messages of type 1 and 2 are exchanged between hosts of the LAN. The other types of messages are exchanged only between a host and the AKD.

4.5 The Protocol Description

Every host on the network first chooses an integer N, a secret K and a random number R. In practice, the first 128 bit of the private key of the host suffices as K. The significance and the choice of N will be clear later on.

The host now computes a signature[2]

$$S(IP, MAC, N, H^{N+1}(K||R), T)$$

Where T is the timestamp. This signature S will be useful for a period $(N+1)T_e$ where T_e = the cache entry validation time (usually equal to 20 min), i.e., the signature will be valid for a time T_e and can be renewed upto N times using N one time passwords. This signature is stored by the host and will be used later in answering ARP request.

4.5.1 Cache Entry Creation

Consider the scenario when a host H_i needs to know the MAC address of H_j. H_i checks its cache and consequently finds no entry for H_j. It queries H_j with an ARP request having field type=1, i.e., a new entry creation request is sent to H_j.

H_j now computes

$$n = (N+1) - \lfloor (t - T) / T_e \rfloor$$

Where t is the current time and the function $\lfloor x \rfloor$ is the floor function returning the largest integer smaller than its argument e.g., $\lfloor 3.9 \rfloor = 3$. Informally, $\lfloor (t - T) / T_e \rfloor$ is the number of 20 minute time slots passed after the signature S was computed.

H_j now calculates $H^n(K||R)$, i.e., the $(N+1-n)$th one time password. Note that if the time elapsed between the ARP request and the signature generation is less than T_e (20 min), $n = N+1$ and the one time password is $H^{N+1}(K||R)$ (already specified in the signature). Hence this can be seen as the zeroth one time password. Further $(N+1-n)$ cannot be negative or less than 1 as when it reaches 1, a new signature with new R should be computed.

H_j now sends an ARP reply to H_i with type=1 and data = $S(IP, MAC, N, H^{N+1}(K||R), T)$, n, $H^n(K||R)$, i.e., data contains the signature S and the computed one time password. Upon receipt of the ARP reply, H_i verifies the signature using the public key of H_j (if it does not already have the required key, it obtains it from the AKD, see the next sub-section), validates the one time password supplied using N and $H^{N+1}(K||R)$ given in signature S and computes $t = T+T_e*(N+1-n)$. If t is within time T_e (i.e., 20 min) from the current local clock time, the reply is accepted and the following five tuple entry is created for H_j

$$<IP, MAC, n, H^n(K||R), t>$$

The last three values are stored in order to avoid having H_j send the signature S everytime and to avoid the overhead of signature verification in each ARP reply.

[2] This signature S represents the data IP, MAC, N, $H^{N+1}(K||R)$, T as well as the digital signature on it

4.5.2 Cache Entry Refreshment

Now consider the scenario in which H_i requires the MAC address of H_j and find a cache entry for it. It checks the stored values n and t. If t is within time T_e (i.e., 20 min) from the current local clock time, the cache entry is considered to be good. No ARP request is sent and the stored MAC address is used. Otherwise, it computes the value t+nT_e and compares it with the local clock time. If the local clock time is more than t+nT_e, this corresponds to (N+1–n) being less than 1, i.e., it corresponds to requiring one time password with index more than N which is non-existent (only N one time passwords exist). This means that the parent signature of the existing cache entry should have expired beyond renewal. At this point, the cache entry is deleted and an ARP request is sent with type=1, i.e., for the creation of a new cache entry. H_j replies as described in the previous subsection.

Finally, if the local clock time is less than t+nT_e, a request of type=2, i.e., a request for entry renewal is sent. H_j now computes the new n and the required one time password $H^n(K||R)$ as described in the previous sub-section. The only difference lies in the ARP reply which is of type=2 and the signature S is not included in the data to be sent. The ARP renewal reply can be validated by computing the hash of the sent one time password (n'-n) number of times and comparing with the stored password, where n' is the value of new n included in the ARP reply. If they match, a check on the quantity t+T_e*(n'–n) is performed which should be with time T_e (i.e., 20 min) from the current local clock time. If the check succeeds, t is replaced with t+T_e*(n'–n), $H^n(K||R)$ with $H^{n'}(K||R)$, n is replaced with n' and the ARP reply is accepted.

Now, we discuss the issue when H_j changes its MAC address. In that case, H_j just needs to discard its old computed signature and should recompute it using the new MAC, a new R and correspondingly all new values including T. The entry creation requests proceeds without any change. In case of entry renewals, however, since the one time password supplied in this case will not be correctly validated by H_j storing the old entry corresponding to the old signature and the old MAC, H_i discards its stored cache entry and sends a new ARP request with type=1.

4.5.3 A Rough Performance Comparison with SARP

On the contrary to S-ARP, our scheme requires a constant number of digital signatures per unit time irrespective of the number of ARP entry creation/renewal request received, e.g. with N=100 and T_e = 20 min, we require only one digital signature computation for a time period of t = (N+1)T_e = 33.6 hours. Thus considering a period of one month (30 days) with an average of one ARP request per second, our scheme requires (30*24)/33.6 < 22 signature computations while S-ARP requires 259,200 signature computations. Given that signature computation is about 10,000 times slower than the computation of a hash function [19], it is easy to see that our scheme dramatically improves the performance of the system. Note that our scheme is still exactly as secure as S-ARP.

4.6 Key Management

Key management in our protocol is pretty much the same as in S-ARP [12]. Note that special care is required to be taken when dealing with dynamically assigned IP addresses. Hence, we consider key management in networks with statically and dynamically assigned IP addresses separately.

We will use the following notations in this section:

AKD	Authoritative Key Distributor
$\boldsymbol{H_i}$	Generic host *i*
Rq(a)	Request for object *a*
Rp(a)	Reply carrying object *a*
T	Local clock Time-stamp
$\boldsymbol{A_H}$	Host *H*'s IP address
$\boldsymbol{M_H}$	Host *H*'s MAC address
$\boldsymbol{P_H}$	Host *H*'s Public Key
$\boldsymbol{S_H(x)}$	Message x digitally signed by host *H*

4.6.1 Static Networks

In such networks, the mapping between the keys and the IP addresses is static. Hence, when a host joins the network for the first time, a key pair and an IP address is assigned to it and inserted into the AKD repository. Now consider that a generic host *i* broadcasts an ARP request to find host *j*'s MAC address and upon receiving the reply finds that it does not have the public key of host *j*. Host *i* then contacts the AKD to request host *j*'s public key. AKD then sends the required key in a digitally signed message

The sequence of messages exchanged is as follows.

$$\boldsymbol{H_i \rightarrow AKD}: \quad Rq(PH_j)$$
$$\boldsymbol{AKD \rightarrow H_i}: \quad S_{AKD}(Rp(PH_j) \,||\, H_j \,||\, T)$$

Now, since the public key used for verifying the host's signatures has been securely released by the AKD and the private key corresponding to that public key is known only to the legitimate host, an attacker cannot produce a valid signature for an IP address other than its own. Thus an attacker can no longer send valid malicious ARP replies to poison a host's cache.

Here another possible way of key management is to provide digitally signed certificates to each host containing the mapping between its IP address and the public key. In this case, we require a CA instead of AKD and no active participation of CA in the protocol would be required. A third type of ARP request can be created for which the reply would contain the issued certificate. But such a mechanism will have to deal with intrusions like certificate revocation in case of key compromise. Hence we choose to stick with the option of AKD. Contrary to the public key infrastructures where certificates are used, we see no problem in having an online AKD in our case for the sake of ARP. Any host on the network as identified by the network administrator could act as AKD. Another reason for this choice is the possibility of dynamic networks (discussed next) where such certificates are not possible as IP addresses are dynamically assigned.

4.6.2 Dynamic Networks

In such a network, a DHCP server dynamically assigns IP addresses to the hosts. Since the IP address of a host is not fixed, keys cannot be bound to IP addresses at generation time.

If an organization deploys a secure DHCP server with secure ARP, only well known hosts that have been enrolled in the system and authorized in some way can enter the LAN. Enrollment procedure is a one time activity which takes places when the host joins the network for the first time. Enrollment involves generating a key pair and the corresponding certificate. In this case, IP field of the certificate is empty. AKD manually inserts this certificate into the repository with null IP address and the corresponding public key, using a secure channel.

When a host *H* joins the LAN, it requests the DHCP server to assign to it an IP address. In order to allow the DHCP server and the AKD to identify it and validate the request, *H* timestamps and signs the request and sends its public key along with the request. Before assigning an IP address to *H*, the DHCP server contacts the AKD to verify whether *H* is authorized to be added to the LAN, i.e., if *H*'s key is in the AKD repository and is valid, and to inform the AKD of the IP address the host will be assigned. The message is signed by the DHCP server and includes the supplied public key along with the proposed IP address. The AKD searches its database for the given public key and replies to DHCP with a signed ACK or a NACK along with some other fields to prevent replay attacks. If the response from the AKD is ACK, the DHCP server proceeds with the assignment of the new IP address to *H*, while the AKD updates *H*'s entry in the repository binding *H*'s new IP address to *H*'s key. Else, the DHCP server will not release a new IP to the host. The message exchange sequence in case of a positive response from the AKD is as follows:

$$
\begin{array}{ll}
\boldsymbol{H \rightarrow DHCP}: & P_H \,||\, S_H(\text{DHCP request} \,||\, T) \\
\boldsymbol{DHCP \rightarrow AKD}: & S_{DHCP}(P_H \,||\, A_H \,||\, T) \\
\boldsymbol{AKD \rightarrow DHCP}: & S_{AKD}(ACK \,||\, P_H \,||\, A_H \,||\, T) \\
\boldsymbol{DHCP \rightarrow H}: & S_{DHCP}(\text{DHCP reply} \,||\, A_H \,||\, T)
\end{array}
$$

5 Conclusion

ARP cache poisoning occurs due to lack of message authentication since any host on the LAN can spoof ARP replies containing malicious IP to MAC mapping. There is no satisfactory solution to cache poisoning since all the proposed solutions are either insecure or have unacceptable penalties on system performance.

We propose a new solution to the problem of ARP cache poisoning. Our solution is based on an efficient combination of digital signatures and one time passwords based on hash chains. Digital signatures are almost eliminated in the sense that the system requires less than one digital signature per day. Thus, the performance of our protocol is significantly better than S-ARP which requires digital signature computation for each ARP reply. Further, we do not compromise security at any point and the security of our scheme is the same as S-ARP. Hence, our scheme is efficient as well as secure at the same time.

References

1. M. Barnaba. anticap. http://cvs.antifork.org/cvsweb.cgi/anticap, 2003.
2. B. Fleck. Wireless access points and arp poisoning [online document]. Available at http://www.cigitallabs.com/resources/papers/download/arppoison.pdf.
3. F. Hunleth. Secure link layer. http://www.cs.wustl.edu/fifhunleth/projects/projects.html.
4. M. Laubach. Classical IP and ARP over ATM. RFC 1577, 1994.
5. A. Ornaghi and M. Valleri. A multipurpose sniffer for switched LANs. http://ettercap.sf.net.
6. D. C. Plummer. An ethernet address resolution protocol. RFC 826, 1982.
7. D. Song. A suite for man in the middle attacks. http://www.monkey.org/fidugsong/dsniff.
8. R. W. Stevens. TCP/IP Illustrated, vol 1. Addison Wesley, ISBN 0-201-63346-9, 2001.
9. I. Teterin. Antidote. http://online.securityfocus.com/archive/1/299929.
10. R. Wagner. Address resolution protocol spoofing and man in the middle attacks. http://rr.sans.org/threats/address.php, 2001.
11. S. Whalen. An introduction to arp spoofing [Online document]. Available at http://packetstormsecurity.nl/papers/protocols/intro_to_arp_spoofing.pdf, 2001.
12. Bruschi, D., Ornaghi, A., Rosti, E., S-ARP: a Secure Address Resolution Protocol, in Proceedings of 19th Annual Computer Security Applications Conference (ACSAC), 2003.
13. Leslie Lamport, "Password Authentication with Insecure Communication", Communications of the ACM 24.11 (November 1981), pp 770-772.
14. N Haller, "The S/KEY One-Time Password System", Proceedings of the ISOC Symposium on Network and Distributed System Security, pp 151-157, February 1994.
15. A. Stemmer, "CAMs Enhance Network Performance", System Design [Online document], Jan. 98, Available HTTP: http://www.eedesign.com/editorial/1998/systemdesign9801.html
16. http://cert.uni-stuttgart.de/archive/vulndev/2002/ 01/msg00295.html
17. Whalen, S. H., Towards Layer 2 Authentication: Preventing Attacks based on Address resolution Protocols Spoofing., 2003. http://wp.netscape.com/eng/ssl3/draft302.txt, 2002.
18. S. Convery, "Hacking Layer 2: Fun with Ethernet Switches", Blackhat [Online document], 2002, Available HTTP: http://www.blackhat.com/presentations/bh-usa-02/bhus-02-convery-switches.pdf
19. S. Micali "NOVOMODO: Scalable Certificate Validation and Simplified PKI Management" First Annual PKI Research Workshop - Proceeding, April 2002.
20. Hacking UNIX 2003, a tutorial for performing various attacks including ARP poisoning attack, on UNIX systems. Available at http://duho.cjb.net.
21. M. V. Tripunitara and P. Dutta. A middleware approach to asynchronous and backward compatible detection and prevention of arp cache poisoning. In Proc. 15th Annual Computer Security Application Conference (ACSAC), pages 303-309, 1999.