

# TARP: Ticket-based Address Resolution Protocol

Wesam Lootah, William Enck, and Patrick McDaniel

*Systems and Internet Infrastructure Security Laboratory  
Department of Computer Science and Engineering  
The Pennsylvania State University  
{lootah, enck, mcdaniel}@cse.psu.edu*

---

## Abstract

IP networks fundamentally rely on the Address Resolution Protocol (ARP) for proper operation. Unfortunately, vulnerabilities in the ARP protocol enable a raft of IP-based impersonation, man-in-the-middle, or DoS attacks. Proposed countermeasures to these vulnerabilities have yet to simultaneously address backward compatibility and cost requirements. This paper introduces the *Ticket-based Address Resolution Protocol* (TARP). TARP implements security by distributing centrally issued secure IP/MAC address mapping attestations through existing ARP messages. We detail the TARP protocol and its implementation within the Linux operating system. We also detail the integration of TARP with DHCP for dynamic ticket distribution. Our experimental analysis shows that TARP improves the costs of implementing ARP security by as much as two orders of magnitude over existing protocols. We conclude by exploring a range of operational issues associated with deploying and administering ARP security.

*Key words:* Network security, ARP security

---

## 1 Introduction

The *Address Resolution Protocol* (ARP) [40] is the glue that holds together the network and link layers of the IP protocol stack. The primary function of ARP is to map IP addresses onto hardware addresses within a local area network. As such, its correctness is essential to proper functioning of the network. However, like other protocols within IP, ARP is subject to a range of serious and continuing security vulnerabilities [11,12]. Adversaries can exploit ARP to impersonate hosts, perform man-in-the-middle attacks, or simply launch a Denial of Service attack. Moreover, such attacks are trivial to perform, and few

countermeasures have been widely deployed. Current network environments present two central design challenges for ARP security. Firstly, the solution must not require ARP be discarded. The deployed base of IP is large and diverse enough that replacing any major component of the IP protocol stack is not only technically challenging but also cost prohibitive. Secondly, the costs of implementing ARP security must be minimal. Resource constrained devices and already computationally loaded hosts cannot afford to budget large amounts of resources for ARP security. Any solution that would demonstrably change the performance profile of ARP will not be adopted. The primary reason that proposed solutions [25,13,22,27] have not been widely deployed is that they have yet to simultaneously address these two requirements.

In this paper, we introduce the *Ticket-based Address Resolution Protocol* (TARP) [28]. TARP implements security by distributing centrally generated IP/MAC address mapping attestations [44,5]. These attestations, called *tickets*, are given to clients as they join the network and are subsequently distributed through existing ARP messages. Unlike other popular ARP-based solutions, the costs per resolution are reduced to one public key validation per request/reply pair in the worst case. As such, TARP is a feasible approach for the diverse assortment of existing network capable devices. We provide a detailed description of the protocol design and integration with the *Dynamic Host Configuration Protocol* (DHCP) [17] for dynamic ticket distribution, as well as its implementation within the Linux operating system. Our experimental analysis shows that TARP retains compatibility while reducing the request costs by as much as two orders of magnitude over existing protocols. We explore a range of crucial operational issues including revocation and incremental deployment and show how TARP can be deployed with limited administrative oversight.

Note that TARP embodies a central design trade-off. Ticket generation costs grow at the linear inverse of the ticket’s lifetime. The ticket lifetime dictates the vulnerability to replay attacks<sup>1</sup>. Hence, administrators can directly control cost and security through the selection of ticket lifetime. The ability to balance between these competing factors is a central benefit to TARP’s design. We explore the management of this tradeoff throughout and reflect on the necessity of such compromises in the practical use of security technologies.

Security in resolution services remains an open problem. Whether resolving domain or hostnames [21,7,8], or claiming address ownership [44,5], one needs to authenticate the contents and freshness of received data. This work represents a new point in the design space of these services. As such, it can be used to determine the specific costs and advantages of resolution services. In partic-

---

<sup>1</sup> We consider an alternate design in a section on revocation in which we address replay vulnerabilities.

ular, our practical analysis indicates that for certain kinds of resolution, great performance gains can be achieved by slightly relaxing security requirements.

We begin in the next section by providing background on ARP and consider the vulnerabilities inherent in its current design. Section 3 overviews past efforts of securing ARP and other related works. Section 4 details the TARP architecture and its operation within local networks. Section 5 outlines the implementation of TARP within Linux. Section 6 explores the performance of TARP. Section 7 considers several operational issues associated with the use of TARP. Section 8 concludes.

## 2 Background

The *Address Resolution Protocol* (ARP) [40] is used by hosts to map IP addresses onto *Medium Access Control* (MAC) link layer addresses. The resulting address associations are used to direct packet delivery within the physical local network.

Every packet in an IP network must be delivered to some interface in a local network. Those whose destination IP addresses are external to the local network (as determined by the subnet mask) are delivered to the subnet gateway, while packets destined for the internal network are delivered directly. Whether the destination address is local or external, the IP address must be mapped onto a MAC address. ARP resolution performs a distributed lookup via a simple broadcast request followed by a unicast response. The querying host sends the request to the local broadcast address. According to the protocol, only a host assigned to the requested IP address should reply with its local hardware address. This reply, containing both the requested IP address and associated MAC address, is sent directly to the querying host. The host caches the association, which expires and is evicted at a later time as determined by local policy. Once evicted, the host repeats the request, cache, and eventual ejection. While the cache hold time for a response is undefined in the protocol specification, many implementations set the expiration to approximately 20 minutes, with the option of resetting the expiry timer after each use [13].

Since hosts implicitly trust the address associations residing in the ARP cache, if an adversary can influence these values, the host can be manipulated into sending packets to the wrong hardware address. The lack of authentication of address association data leaves hosts susceptible to reply spoofing and cache entry poisoning, commonly referred to as *cache poisoning*. In fact, freely available tools are designed to exploit these vulnerabilities [45].

Most IP protocol stacks are designed to ignore unsolicited ARP replies. How-

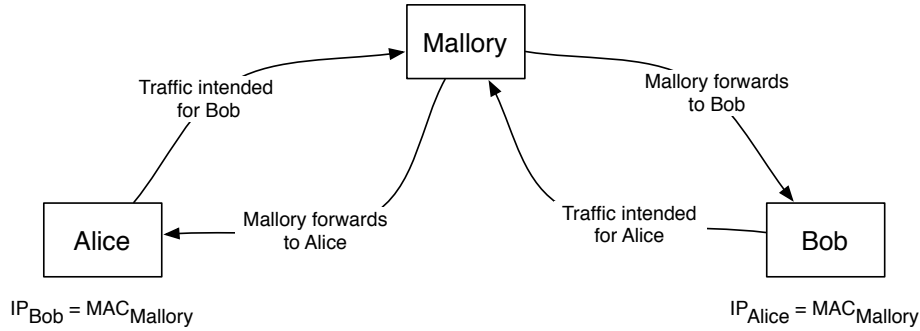


Fig. 1. Example of a Man-in-the-Middle attack in progress.

ever, this does little to prevent cache poisoning. An adversary can coerce a host into requesting a specific address by spoofing an ICMP ping message. The spoofed message contains the targeted IP address, requiring the host to resolve the MAC address to reply. By spoofing replies and poisoning the cache of network hosts, an adversary can perform both *Denial of Service* (DoS) and *Man in the Middle* (MITM) attacks [20]. Although such attacks were identified a decade and half ago [11], they still exist today [12].

Cache poisoning can be used to mount various types of DoS attacks. In the simplest case, the adversary replaces the MAC address of a particular host with another value. When the victim attempts to communicate with that remote host, all traffic is sent to the wrong MAC address. This effectively denies service to the remote host. If this remote host happens to be the gateway, the host will be unable to communicate with hosts outside of the subnet. Finally, if the adversary knows the IP address of all nodes on the subnet, cache entries can be crafted so that the victim cannot communicate with any remote hosts.

While DoS is a serious concern, cache poisoning resulting in a MITM attack is more dangerous. This attack, as shown in Figure 1, not only allows the adversary to insert messages into the communication channel, but more importantly, it often goes undetected. Furthermore, cache poisoning used in this manner allows eavesdropping even on a layer-2 switch. In order to launch this attack, the adversary must effectively manipulate the caches at both ends of a conversation. Once both ends believe the adversary is the correct remote destination, manipulating packet streams is trivial.

### 3 Related Work

#### 3.1 External solutions

Several attempts have been made to address the security issues posited above. Some attempts use methods external to the protocol. These methods do not use cryptography and do not modify the protocol in any way. For example, it has been proposed that hosts can be configured with static ARP entries [1]. Although this maybe a viable solution for very small and static networks, it is completely intractable for large dynamic networks due to its huge administrative overhead.

Port security [14], a security feature available in recent switches, restricts the use of physical ports to configured MAC addresses. This approach only prevents certain kinds of MAC hijacking, but does nothing to prevent MITM attacks. Hence, it is only a partial (and in many ways limited) solution.

*Dynamic ARP Inspection* (DAI) [16], a security feature available on some recent network switches, is another solution external to the protocol. DAI provides security by preventing invalid or malicious ARP packets from being forwarded on the network. When DAI is enabled on a network switch, the switch inspects ARP packets before forwarding them. ARP packets are compared to a database of valid IP-to-MAC address bindings. If an ARP packet contains a valid binding, it is forwarded; otherwise, it is dropped. Clearly, key to the proper operation of DAI is the IP-to-MAC address binding database. A system administrator can maintain this database manually, or preferably it can be built using DHCP snooping [15]. DHCP snooping, also a security feature, can be used in concert with DAI. With DHCP snooping, the port on which the network DHCP server is connected is labeled “trusted,” while all other ports are labeled “untrusted.” Only DHCP server packets that originate from a “trusted” port are forwarded, while all other DHCP server packets are dropped. When DHCP snooping is enabled on a switch, it acts as a firewall for DHCP traffic. It also builds and maintains a database of IP-to-MAC address bindings from information in DHCP packets. This database can then be used with DAI to provide security against ARP-based attacks.

Although DAI with DHCP snooping can provide effective security against ARP based attacks, it is not widely available on commodity layer-2 switches. Moreover, the overhead of inspecting packets at layer-2 can be prohibitive. The overhead of deep inspection increases latency and has a negative impact on throughput. It is also important to note that DAI and DHCP snooping are not available for wireless environments, where physical port security is not available.

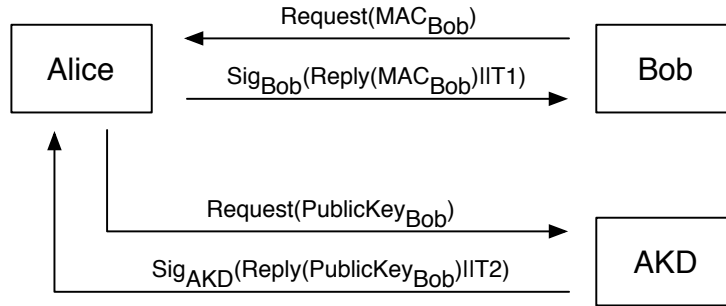


Fig. 2. S-ARP address resolution. Alice broadcasts a request for Bob’s MAC address. Bob sends Alice a signed reply. Alice requests Bob’s public key from the AKD. The AKD responds with a signed reply.

Other solutions attempt to detect misbehavior, rather than prevent it. ARP-Watch [27], a network-level detection device, detects malicious ARP packets by monitoring MAC/IP address pairings occurring on a subnet. Conversely, host-level detection services differ in that each host on the network attempts to detect malicious messages arriving at the local interface [46]. This is achieved by detecting duplicate and/or unsolicited ARP packets. Detection techniques are punitive by definition, and hence are of limited utility in many environments.

### 3.2 Cryptographic Solutions

A number of cryptographic protocols have targeted security issues in ARP. In the *Secure Link Layer* (SLL), all link layer traffic is authenticated and encrypted. While this prevents authorized hosts from injecting malicious messages, it does not prevent authorized, but untrustworthy hosts, from injecting malicious messages. SLL is an example of a solution that provides message authenticity but no proof of IP address ownership.

In another approach, Gouda and Huang [22] propose the Secure Address Resolution Protocol. A secure server in this protocol shares secret keys with each host on a subnet. The server maintains a database of IP-to-MAC address mappings that is updated periodically through communication with each host. All ARP requests and replies occur between a host and the server, and replies are authenticated using the shared pair keys. Note that the server represents a singular point of failure and congestion, which make it a poor match for most networks.

Another ARP security protocol, S-ARP [13], depicted in Figure 2, uses asymmetric cryptography. Hosts use self-generated public/private key pairs certified by a local trusted party. Each host registers its public key with the *Authorita-*

*tive Key Distributor* (AKD) server. The server's public key and MAC address are also securely distributed to all hosts during a bootstrapping process. S-ARP requests proceed as normal ARP requests. However, S-ARP replies are signed by the sender's private key. Upon receiving a reply, the signature is verified using the sender's public key. If the receiver does not have the sender's public key, or if the signature cannot be verified by the keys currently in its key ring, the public key of the sender is requested from the AKD. The AKD sends it to the requesting host in a signed message. If the new public key verifies the signature, the reply is accepted and the public key is cached; otherwise, it is rejected. To avoid replay attacks, messages are time-stamped and synchronization messages are exchanged with the AKD. S-ARP requires, at minimum, a single signature generation and verification per address resolution. As illustrated in Section 6, this cost can be significant.

### 3.3 Neighbor Discovery

The *Neighbor Discovery Protocol* (NDP) [37], ARP's counterpart in IPv6, is used to discover other nodes on the same link as well as determine their link-layer addresses. NDP is also used to find routers, and to maintain reachability information about paths to active neighbors. Although NDP was developed a decade after ARP's security problems were identified, it still suffers from the same fundamental security flaws: a lack of message authentication and proof of address ownership. The original NDP standard [37] suggests the use of IPsec to add message authentication. This approach has a bootstrapping problem, as a protocol such as IKE is needed to automate the creation of security association. However, since IKE is a higher layer protocol that depends on NDP, it cannot be used to secure NDP. Therefore, all that remains is manual key distribution, which is intractable for large and dynamic networks.

To address the security problems with NDP, Kempf proposed using *Address Based Keys* (ABK) [25], which uses Identity-Based cryptography. In this scenario, IP addresses are used as public keys. However, contemporary identity-based systems require one or more heavyweight cryptographic operations per signature generation or validation. Hence, their cost is prohibitive for many resource poor devices.

*Cryptographically Generated Addresses* (CGA) [9] has also been proposed to address the security in NDP. In CGA, the host address is partially based on its public key. The 62-rightmost-bits of an address are generated by hashing the host's public key, while the remaining leftmost bits represent the network prefix. This provides immediate binding between a host's IP address and its public key, which allows a host to claim ownership of an IP address by demonstrating that it possesses the private/public key pair associated with that

IP address.

CGA has been proposed for IPv6 where the length of an IP address allows for a large enough number of bits to make brute force attacks ineffective. In IPv4, the length of an IP address is far too small. Even if the full 32-bits are cryptographically generated, it still would not provide adequate security against a brute force attack. With current technology, a well financed attacker can easily generate an IP-to-Public key lookup table for all possible IPv4 addresses.

The above mentioned solutions are either partial or limited in scope. None of the solutions simultaneously address both the compatibility and performance requirements of current networks. As we will show in the following section, TARP successfully achieves resilience to cache poisoning and compatibility with ARP, at virtually no cost.

#### 4 A Ticket-Based Approach

The security flaws in ARP stem from the following facts: 1) ARP messages lack guaranteed integrity and authenticity; and 2) ARP messages do not provide proof of IP address ownership. Therefore, adding authentication alone to ARP messages does not solve the problem. Authentication must also be combined with a means to prove address ownership.

We address these requirements through the *Ticket-based Address Resolution Protocol* (TARP) [28]. TARP implements security by distributing centrally generated attestations [44,5]. These attestations, called *tickets*, authenticate the association between IP and MAC addresses through statements signed by the *Local Ticket Agent* (LTA)<sup>2</sup>. Each ticket encodes a validity period represented as a start time and an expiration time. Of course, the use of time values assumes some form of loose clock synchronization between the issuing LTA and the validating clients. Such synchronization is a common requirement for many protocols, and devices for its enforcement are well known [34].

To securely perform address resolution using TARP, a host broadcasts an ARP request. The host with the requested IP address sends a reply, attaching a previously obtained ticket. The signature on the ticket proves that the LTA issued it, i.e., the IP-to-MAC address mapping is valid (or at least was at the time of issuance—see revocation below). The requesting host receives the ticket, validating it with the LTA’s public key. If the signature is valid, the

---

<sup>2</sup> The LTA is trusted by all network hosts to make claims about IP-to-MAC address mappings.

address association is accepted; otherwise, it is ignored. With the introduction of TARP tickets, an adversary cannot successfully forge a TARP reply and, therefore, cannot exploit ARP poisoning attacks.

The remainder of this section discusses the protocol in detail and considers revocation. The following notation is used:

$H_i$	A Generic host
Request(x)	Request for object x
Reply(x)	Reply with object x
$Ticket_{H_i}$	$H_i$ 's digitally signed ticket
$MAC_i$	$H_i$ 's MAC address

#### 4.1 The TARP Protocol

The means by which a ticket is created and distributed is dependent on whether the IP address assignments are static or dynamic. In either case, the address resolution exchange consists of a broadcast request and unicast reply as follows:

$$\begin{aligned}
 H_i &\longrightarrow all : Request(MAC_j) \\
 H_j &\longrightarrow H_i : Reply(MAC_j) \parallel Ticket_{H_j}
 \end{aligned}$$

Barring the inclusion of  $Ticket_{H_j}$ , the exchange is identical to ARP. The method in which  $H_j$  obtains  $Ticket_{H_j}$  differs depending on network requirements.

A network administrator may choose to statically assign IP address to hosts. In which case, whenever a host is added to the network, it is configured with the network public key, and network configuration parameters including an IP address, and a ticket, as illustrated in Figure 3. Because the associations are unlikely to change frequently, it may be acceptable to set long ticket lifetimes. However, there are security, performance, and administrative considerations related to the selection of ticket lifetimes. We consider these issues in depth in Section 4.3 below.

In dynamic IP networks, hosts are assigned IP addresses and configuration parameters by a configuration server using the *Dynamic Host Configuration Protocol* (DHCP) [17]. Each host receives a lease on an IP address and sends a renewal request upon expiration. At this time, the DHCP server may or may not reassign the host the same IP address.

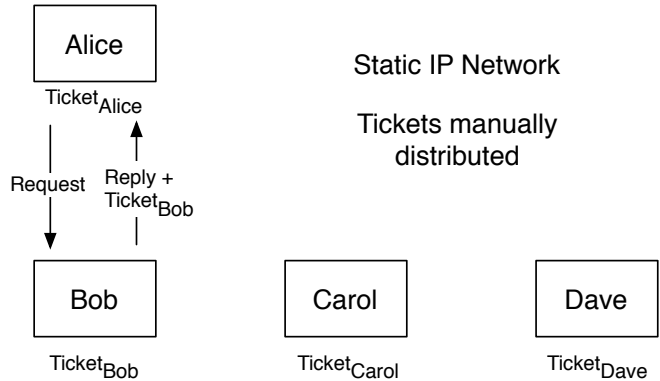


Fig. 3. Static IP Address Assignment - hosts receive TARP tickets during initial setup, and include them with each ARP reply.

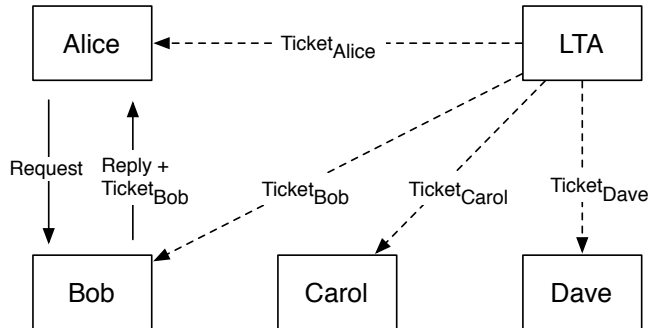


Fig. 4. Dynamic IP Address Assignment - hosts receive TARP tickets during the initial DHCP exchange, and include them with each ARP reply.

In a TARP-enabled dynamic IP network, the DHCP server also performs the functionality of an LTA, as shown in Figure 4. In response to a DHCP request, the server packages a ticket with the configuration information. Accordingly, the ticket expires along with the IP lease. Note that tickets are by definition public; therefore, a secure communication channel is unnecessary. Having the DHCP server play the role of LTA eliminates the need for additional ticket distribution messages, hence maintaining simplicity of protocol design. Additionally, using this method of distribution is logical, as DHCP was designed to distribute configuration parameters.

A host requires the LTA's public key in order to verify tickets. Key distribution is most secure if performed out of band. Another option, although not secure, is Key distribution through assertion and user acceptance, similar to that in the Secure Shell (SSH) protocol [47]. Unfortunately, this allows an adversary new methods of attack. For this paper, we only consider out of band, manual, distribution of the LTA's public key.

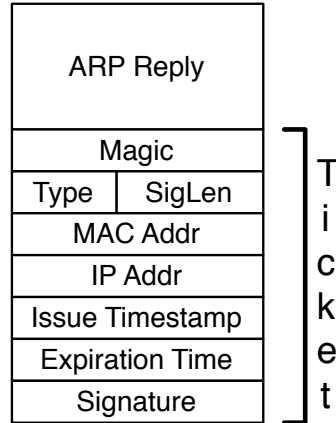


Fig. 5. TARP Reply Packet Format - the TARP signature covers all fields from the Magic through the expiration time.

#### 4.2 Ticket Format

Maintaining backwards compatibility with ARP is crucial for the adoption of any enhanced address resolution protocol. Compatibility is achieved by integrating the ticket into the ARP reply; no changes need take place to the request. As shown in Figure 5, the ticket is appended as a variable length payload, with the ticket header modified accordingly.

The *Magic* field in the ticket header is used to distinguish a TARP reply from an ARP reply. In a TARP reply, the magic field is set to  $0x789a0102$ <sup>3</sup>. Since TARP has only one message type, the *Type* field actually designates the cryptographic algorithm<sup>4</sup>. The *SigLen* field indicates the signature length. The remaining fields contain information required to ensure proper operation. The *MAC Addr* and *IP Addr* fields create the address association. The *Expiration Time* field indicates how long the ticket is valid. *Issue Timestamp* field indicates when the ticket was generated and is used for ticket revocation as discussed below.

It is possible to imagine tickets generated with very short validity periods. In this case, the overhead of TARP approaches the overhead of S-ARP. In fact, we can think of TARP’s validity period as a generalization of the freshness timestamp included in S-ARP.

The ticket validity period is a key element in the design of TARP. It allows the cost of ticket generation to be amortized over the lifetime of a ticket. This

<sup>3</sup> The magic field appeared in S-ARP, and we use it for a similar purpose.

<sup>4</sup> As discussed in Section 5, our implementation currently uses 1024-bit RSA, but other key sizes and algorithm may be used as appropriate and desirable.

concept is not unique to TARP. DNS Security Extensions (DNSSEC) [6–8], uses a signature inception and expiry information in *signature resource records* (SIG RR). The signature inception and expiry provide a validity period for SIG RRs. It is not surprising that both DNSSEC and TARP make use of validity periods for signed data, as both protocols’ primary objective is to add security to an address resolution protocol while keeping overhead to a minimum.

### 4.3 Revocation

A reality of current networks is the fact that IP/MAC address associations can change; dynamic bindings (e.g., DHCP) or changes in network configuration can occur before a ticket expires. To be secure, one must provide a *revocation* mechanism that securely notifies clients which tickets are no longer valid. Historical studies of revocation have sought to limit the cost of notification, e.g., CRLs and other data structures [24,33,4,38,26], limit notification latency, e.g., OCSP [36], or provide frameworks for trading off security guarantees and semantics [30,4,23].

Revocation speaks to the central tradeoff of TARP. Because a revoked ticket may be replayed at any time prior to its expiration, administrators may be tempted to keep the lifetimes short. However, ticket issuance costs grow at the linear inverse of the ticket lifetime. The ability to calibrate the balance between these competing factors through the selection of ticket lifetimes is a central benefit to its design.

The simplest method of handling revocation is to issue certificates that are only valid for a short time. This is similar to the *short lived certificates* suggested by Ellison et al. in the SPKI/SDSI system [19,42]. Because the tickets are only valid for a short time, the vulnerability to replay is limited and no notification is necessary. Note that a window of vulnerability to replay also exists in S-ARP. The window that is equal to the cache hold time of the ARP reply. Users of TARP can provide similar window by setting the lifetime of the ticket to the ARP cache hold time. However, the burden of the creating the tickets is on the LTA, rather than on the hosts themselves. We experimentally explore the costs of the ticket creation and validation in Section 6.

ARP associations are long lived in networks where IP addresses are assigned manually. For this reason it may be advantageous to create tickets whose lifetimes are essentially infinite for these static associations<sup>5</sup>. In those rare cases where mappings change, one can revoke through re-issuance; all clients would

---

<sup>5</sup> The expiration time field is a 32-bit value. When set to its maximum value, the ticket will expire in 2038.

only use the ticket with the latest expiry timestamp. This “latest ticket wins” approach would be vulnerable to active attacks in which the adversary can block delivery of the new ticket. Such attacks represent a powerful adversary within the local area network, and may signal larger and more serious problems. Hence, the risk may be acceptable for many environments.

The most secure solution is to implement a separate revocation service. Such solutions range from the distribution of simple signed certificate revocation lists [24] to instantaneous online verification of ticket validity [36]. Note that simple solutions like CRLs are likely most appropriate, as the costs of the complex ones would eclipse the costs of securing ARP. Hence, we expect that simple, low cost solutions will be used in all networks but those with the highest security requirements. We defer further discussion of the design tradeoffs of revocation services to the relevant literature [35,31].

An important question is how to recover in the presence of compromise of the LTA. This issue is similar to CA recovery in PKI systems. Unlike many PKI deployments, all TARP clients serviced by a LTA are likely to be under a single administrative domain. Hence, it is reasonable to expect that each client can be manually configured with a new certificate as needed. Larger domains may employ techniques to reduce the impact of LTA compromise, e.g., key-splitting [29], issue and revoke LTA keys through local certificate management services, and may use automated management tools for the distribution of LTA signing keys.

#### *4.4 Attacks against TARP*

Networks implementing TARP are vulnerable to two types of attacks: active host impersonation, and Denial of Service through ticket flooding. In the following we discuss these attacks and show how they can be mitigated.

##### *4.4.1 Active host impersonation*

An active adversary that can block all communication between two hosts can impersonate its victim by spoofing its MAC address and replaying a captured ticket. While this attack is present in the ARP, with TARP, the adversary can only impersonate the victim as long as the ticket is valid. Furthermore, a variant of this attack is present in any solution that uses caching. Fortunately, this attack can be mitigated by using a layer-2 switch with port security, thereby preventing MAC address spoofing.

#### 4.4.2 Ticket flooding

An adversary can also take advantage of the cost imbalance between generating a TARP reply and processing it. Exploiting this, a DoS attack can be launched by flooding the victim with bogus TARP replies. These bogus replies are trivial to generate, hence allowing the adversary to easily send thousands of TARP replies at a cost magnitudes lower than the resulting validation attempts. By flooding a victim with bogus ICMP requests and corresponding TARP replies, the adversary successfully circumvents even a stateful ARP implementation and consumes the victim's CPU resources. As this attack results from ICMP behavior, mitigation requires adaptation of that protocol. An attempt to mitigate the risk of DoS attacks using ticket flooding can be done by detecting such an attack at the host and switching to a mode of operation where a client puzzle [32] is sent with the request. The host receiving the request with the puzzle will have to solve the puzzle and include the answer in the TARP reply. The host that originated the request will check the puzzle solution before validating the TARP ticket. The cost of validating a client puzzle is a fraction of the cost of ticket validation and therefore eliminates the cost imbalance between TARP reply generation and validation. It is important to note that this solution, as described above, introduces a DoS attack by flooding a host with client puzzles. A quick solution to this attack is to limit the number of client puzzles processed per second. Although this rate-limiting solution solves the DoS attack, it still denies the host under attack access to other hosts under ticket flooding attack.

Another approach to solving the ticket flooding attack is to use a layer-2 switch with rate-limiting features. Such switches can limit the rate of ARP packets originating from any or all switch ports.

## 5 Implementation

We have implemented TARP on Linux, version 2.6. The source code is available for download<sup>6</sup>. Our implementation has two primary goals: to demonstrate that TARP indeed works and is compatible with ARP; and more importantly, to measure the overhead of TARP and compare it to the overhead of both ARP and S-ARP.

Our implementation makes use of a number of libraries, including libpcap [3] for packet capture, libnet [43] for packet injection, and OpenSSL [2] for cryptographic operations. Similar to S-ARP, our implementation has two main components: a loadable kernel module, and a userspace daemon. The kernel

---

<sup>6</sup> <http://siis.cse.psu.edu/tools.html>

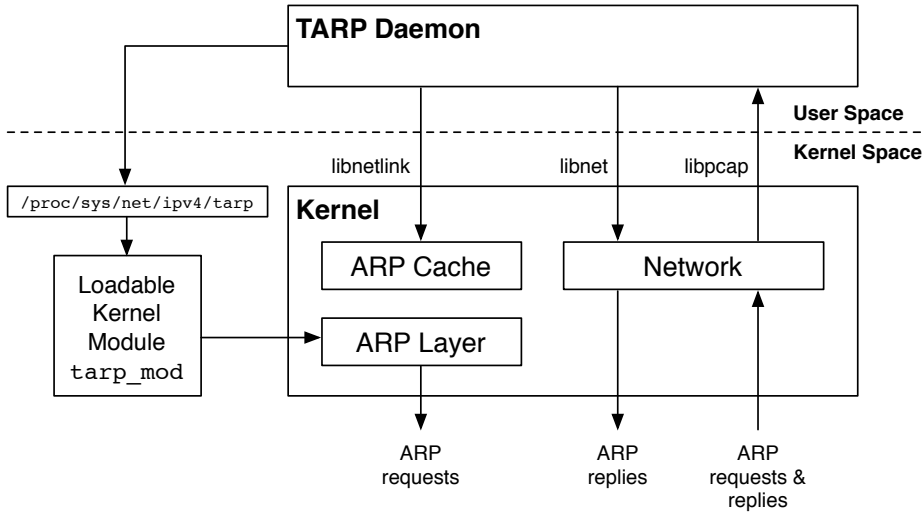


Fig. 6. The TARP implementation architecture (similar to S-ARP)

module provides the proper hooks to disable kernel processing of incoming ARP packets. This allows the userspace daemon, `tarpd`, to capture and respond to incoming ARP packets. By implementing most of the functionality in userspace, we gain better portability and avoid implementing complex cryptography in the Linux kernel.

When loaded, the userspace daemon instructs the kernel module (through the `/proc` file system) to disable kernel processing of incoming ARP packets and waits for ARP packets to arrive. When an ARP packet arrives, it is processed according to its type. If it is a request, a TARP reply is sent. Otherwise, it is a reply, and the source IP address is compared to whitelist entries. If not found, it is treated as a TARP reply and the attached ticket is verified. If the ticket is valid the ARP cache is updated using a netlink socket, and the ticket is cached (in a hash table) to speedup later verifications.

The current implementation of TARP uses RSA with 1024-bit keys. We choose RSA among a number of alternative signature schemes because of its fast signature verification and the availability of highly efficient open source implementations (OpenSSL). We also choose a 1024-bit key size as it is fit-to-purpose, striking a good balance between security and performance.

Our current version of TARP includes an administrative tool to generate the LTA's public/private key pair as well as tickets for manual distribution. We have also implemented an LTA server for dynamic ticket distribution.

During the design phase of the LTA, we have considered two different approaches. In the first approach, a DHCP server does ticket distribution. This requires modifying the DHCP server code to perform LTA functions. The DHCP server sends TARP tickets along with DHCP OFFER messages. The

ticket is included in the DHCP message as a DHCP option. Similarly, the DHCP client is modified to parse and use the ticket embedded in the DHCP OFFER message. Once a DHCP OFFER is accepted and an ACK message is received from the DHCP server, the DHCP client sends the ticket to the TARP daemon. This approach for ticket distribution does not require any specific ticket distribution messages. However, it does require significant modifications to both DHCP server and client code.

In the second approach, neither the DHCP server nor the client is changed. In this approach, the LTA is separated from the DHCP server. However, the LTA is still triggered by the messages exchange between the DHCP server and client. Once a DHCP OFFER message is sent by the server and accepted by the client, the LTA sends a special TARP message. The message is an ARP message with a special opcode (opcode 5). This message contains the ticket; binding the IP address in the DHCP lease with the MAC address of the client for the same IP address lease duration. The TARP process in the client receives this message validates the ticket and uses it for future TARP resolutions.

We choose the latter approach for implementing the LTA. This approach has the advantage of not being dependent on the source or version of the DHCP server running in an environment, making it much more deployable. We believe that an open source TARP aware DHCP server and client are still needed; however, we leave such an implementation for future work.

Our LTA uses the libpcap library to capture DHCP messages (UDP port 67 and 68). The DHCP options in the packet are parsed. Option 53 represents the DHCP message type. It is used to identify DHCP ACK messages. Option 51 represents the lease time in seconds and it is used to set the expiry time of the generated ticket. The ticket IP and MAC address are copied from the fields *yiaddr* and *chaddr* from the DHCP message header. We also use the packet header provided by libpcap to ensure that the DHCP server generated the packet and it was not spoofed by another host<sup>7</sup>.

The LTA uses the same libraries (OpenSSL and libnet) as `tarpd` to generate and send the tarp ticket. Our implementation of the LTA integrates seamlessly with DHCP without the need for changes to the protocol or its implementation.

---

<sup>7</sup> We assume that the LTA process is running on the same host as the DHCP server.

## 6 Performance Evaluation

In order to understand the cost incurred by TARP, we performed two types of measurements: the macro-benchmark indicates the cost seen by an application, the micro-benchmark evaluates the delay of the primary operations. For the macro-benchmarks, we compare our protocol to both ARP and S-ARP.

Our test environment consists of two desktop PCs and included a laptop as the AKD in the S-ARP measurements. The desktops were equipped with 2.8GHz Pentium 4 processors and 1GB of RAM, while the laptop contained a 1.0GHz Pentium 3 processor and 1GB of RAM. All machines ran version 2.6 of the Linux Kernel and were connected via a Gigabit Ethernet switch. Finally, because S-ARP [39] was written for an earlier version of Linux, small updates were required to compile and run it in our environment.

### 6.1 Macro-benchmark

The macro-benchmark observes the round trip time of the three tested protocols: ARP, S-ARP, and TARP. While direct measurement in the kernel is possible, we chose an indirect route, measuring delay at the application level. For this method, ARP is used as a baseline. The overhead is calculated by taking the difference between ARP and both S-ARP and TARP. Since the overhead is the desired result, the indirect method produces the same results as a direct measurement. Additionally, measuring delays from the application level not only reflects real costs, it provides consistent measurement between the tested protocols.

To observe round trip delay from the application level, we used a custom `ping` application to flush the system’s ARP cache after each ICMP echo request/reply pair. This ensured each measurement included the overhead of address resolution. We performed five experiments, each consisting of 1000 ICMP echo requests. These experiments measured the round trip delay for ARP, S-ARP, and TARP with and without caching.

Table 1 summarizes the mean, standard deviation, and median of the recorded measurements for the protocols operating with caching turned on (best case scenario). The overhead was calculated from the mean. As shown, we observed small standard deviations for each protocol. This resulted from the largely controlled test environment.

With caching turned on, S-ARP observes an overhead of approximately 5.4 ms. This is 55 times greater than TARP, which observes only a 98  $\mu$ s overhead. The delay incurred by TARP is essentially unnoticeable, meeting our low overhead

Table 1

Round-trip delay for ICMP echo requests with caching (1000 requests).

Protocol	$\bar{x}$ ( $\mu\text{s}$ )	$\sigma$ ( $\mu\text{s}$ )	Median ( $\mu\text{s}$ )	$\bar{x}$ Overhead ( $\mu\text{s}$ )
ARP	1178.59	259.98	1108	N/A
S-ARP	6579.57	415.99	6535	5401.02
TARP	1276.54	262.47	1206	97.95

Table 2

Round-trip delay for ICMP echo requests without caching (1000 requests).

Protocol	$\bar{x}$ ( $\mu\text{s}$ )	$\sigma$ ( $\mu\text{s}$ )	Median ( $\mu\text{s}$ )	$\bar{x}$ Overhead ( $\mu\text{s}$ )
ARP	1178.59	259.98	1108	N/A
S-ARP	12479.71	571.47	12176	11319.12
TARP	1364.21	253.93	1297	185.62

design requirement.

Table 2 summarizes the worst case performance measurements. Again, the mean was used to calculate the overhead. When caching is disabled, the delay introduced by S-ARP doubles, resulting in an overhead of 11 ms. On the other hand, TARP is two orders of magnitude faster, incurring only 186  $\mu\text{s}$  of overhead. Hence, when signature verification is required, the delay incurred by TARP is virtually insignificant.

In summary, our results show that TARP out-performs S-ARP by at least an order of magnitude in all experiments, and by as much as two orders of magnitude in some cases. More importantly, the results indicate TARP incurs a virtually insignificant overhead. As discussed in our solution criteria, this is vital to the adoption of a secure replacement for ARP.

## 6.2 Micro-benchmarks

Operationally breaking down TARP’s overhead provides insight into how the protocol will perform on different types of devices. TARP message flow begins by requesting an address association. Since the request is identical to that of ARP, no overhead is introduced. When the remote host replies, a ticket is simply appended to a reply. While this requires additional system I/O and network traffic, the overhead is negligible. Upon receiving a TARP reply, a host must verify the ticket signature. This stage requires an asymmetric cryptographic operation and should therefore be investigated. As TARP operates in userspace, cache updates result in additional context switches, slowing down operation. Determining this cost foretells the gain resulting from a kernel based

Table 3

Execution times in microseconds for TARP operations (Average of 100 measurements).

Operation	Average ( $\mu s$ )	$\sigma$
Ticket Signature Verification	119.12	2.00
Update of ARP cache	74.07	7.15
Ticket Generation	4535.36	68.33

implementation. Finally, TARP gains significant performance improvements by amortizing the cost of ticket generation.

Table 3 summarizes the micro-benchmarks. The experimental environment was more controlled than that of the macro-benchmarks, therefore, even with 100 runs, a small standard deviation was achieved. The ticket signature verification consists mainly of a 1024-bit RSA signature verification. This operation is only required when a received ticket does not exist in the cache. The average time of 119  $\mu s$  corresponds directly to the difference between the two TARP variations measured in the macro-benchmark. The cache update also reflects the values measured in the macro-benchmark. If TARP was implemented in kernelspace, 74  $\mu s$  would be virtually eliminated, removing essentially all overhead when tickets are cached. Finally, ticket generation requires 4.5 ms. As the ratio of requests to ticket generations approaches one, TARP performs similarly to S-ARP. This is where the real power of TARP is introduced.

### 6.3 DHCP Integration Micro-benchmarks

We also measured the overhead of TARP associated with ticket distribution. Ticket distribution in our experiment was done by an LTA. The LTA process is triggered by the DHCP message exchange between a DHCP server and client. The LTA extracts the mapping between IP and MAC addresses from the DHCP ACK message, which indicates that an offer was accepted by the client and further acknowledged by the server. The LTA generates a ticket and sends it the client in a special TARP message.

In our experiment we measured the overhead incurred by the LTA for ticket generation as well as the overhead incurred by the client for ticket validation. Table 4 summarizes our results. As shown, the results agree with the micro-benchmark finding in Table 3.

Table 4

Execution times in microseconds for ticket distribution operations (Average of 100 measurements).

Operation	Average ( $\mu$ s)	$\sigma$
LTA Ticket Generation	5383.62	984.95
New Ticket Verification	151.26	33.75

## 7 Discussion

In this section, we briefly discuss TARP with respect to DHCP, interoperability, and Secure Neighbor Discovery.

### 7.1 DHCP

As previously indicated, TARP does not include key and ticket distribution messages. Instead of creating a new distribution protocol, DHCP is used. Clients receiving tickets alongside DHCP replies can readily authenticate the DHCP reply by verifying the signature on the ticket. However, this only provides one-way authentication. In some cases, authenticating clients before distributing DHCP leases and tickets may be required in order to restrict network access or avoid attacks such as IP address pool exhaustion.

Methods for securing DHCP exist. Suggestions include Authentication for DHCP [18], where the protocol has been extended with additional security parameters. Of course, such systems need a way to tie authentication into a central system. Many network installations already have such devices deployed. Applicable back-ends include RADIUS [41], a common authentication database. How and when such authentication is performed is the subject of network policy and influenced by available infrastructure, and hence should be dealt with as operational needs dictate.

### 7.2 Interoperability

While TARP successfully and efficiently prevents cache poisoning, it is useless without a plan for incremental deployment. Mixed networks result in two scenarios of interest to incremental deployment: 1) an ARP host,  $H_a$ , sends an ARP request to TARP-enabled host,  $H_t$ ; or 2) a TARP host,  $H_t$ , sends a request to an ARP host,  $H_a$ .

In scenario 1, when  $H_t$  receives an ARP request, it does not know if  $H_a$  runs

the original protocol or not, because both request packet forms are identical.  $H_t$  proceeds to return a TARP reply.  $H_a$  receives this reply and parses it correctly. This occurs, because to an ARP host, the ticket simply appears as network garbage. Hence,  $H_a$  can successfully resolve  $H_t$ 's MAC address and therefore transmit data.

In scenario 2,  $H_a$  receives a TARP request, which is identical to an ARP request, and replies to  $H_t$  with an ARP reply (no ticket attached). As  $H_t$  cannot verify the address association, it ignores the reply. After time elapses, the higher layer protocols times out. The only barrier keeping the mixed network from functioning is the verification of an ARP reply by a TARP-enabled host. A TARP-enabled host cannot simply accept all ARP replies; this invalidates any security gained from the new protocol. In order to allow address resolution to proceed in scenario 2, TARP supports whitelists. Whitelist entries are one of two types: whitelisted IP ranges, or static ARP mappings.

TARP supports whitelisted IP ranges. This allows a DHCP server<sup>8</sup> to distribute IP addresses from two different pools—ARP hosts, and TARP hosts. Such a configuration may be necessary for a transition to TARP, as it is needed to specify precisely which hosts are participating in the protocol.

These lists can also contain hard-coded MAC and IP address mappings. While currently not implemented, this type of whitelist can be distributed by the network administrator or DHCP server. This allows dynamic configuration of static ARP entries for known devices that do not support TARP. Example devices include embedded hosts such as routers that require vendor support for protocol updates.

Although TARP is designed to interoperate with ARP to facilitate incremental deployment, hosts running ARP are not in any way protected by TARP. Moreover, TARP cache entries referencing whitelisted hosts are also subject to poisoning. In order to achieve the most from TARP all hosts on the local area network should be migrated to TARP.

### *7.3 Secure Neighbor Discovery*

Secure Neighbor Discovery Protocol (SEND) [10] was designed to provide security to NDP. SEND adds a number of options to NDP that together provide message authentication and proof of address ownership. Address ownership is provided by the Cryptographically Generated Address (CGA) parameters option, while message authentication is provided by the timestamp or nonce option along with the RSA signature option.

---

<sup>8</sup> The DHCP server signs the whitelist with the network private key.

In SEND, each host on the network generates its own address by following the process described in RFC 3972. The address includes the network prefix and a host identifier that is generated by taking a hash of CGA public parameters and the public key of the host. This binds the host address to its public key without the need for any public key infrastructure.

NDP resolves addresses in a manner similar to ARP. To learn the hardware address of a target, a host begins by sending a Neighbor Solicitation (NS message) to a multicast address that the target host should be listening to. The target host receives the NS message and replies directly to the source host with a Neighbor Advertisement (NA message). When using SEND, the target host includes a CGA parameters option, a nonce option, and an RSA option in the NA message. The RSA option includes a signature that is generated using the host's private key. The host receiving the reply validates it by first validating the CGA parameters option, which insures that the target address was cryptographically generated using the included public key. Second, the nonce is checked for freshness and finally the signature in the RSA option is verified. If the reply is valid, the host uses the address association to direct messages to the target host.

It is evident from the above that SEND provides security to NDP; however, the overhead of SEND includes an RSA signature generation and verification per address resolution. This makes the overhead of SEND an order of magnitude higher than the overhead of address resolution using TARP.

We propose using tickets to secure NDP. Tickets can be included in NDP messages as a new option. The ticket option includes a timestamp and validity period. A signature is included as a new RSA signature option. The signature covers the address association and the ticket option, and is generated using the LTA's private key. No nonce or timestamp options are needed. By using the new RSA and ticket options with NDP, the cost of signature generation is amortized over the lifetime of the address association.

In the case where CG addresses are used, there is no need for an LTA; signatures are generated using each host's private key. The overhead remains an order of magnitude lower than the current SEND protocol.

## 8 Conclusions

ARP is essential to the proper operation of IP networks. However, the lack of authentication and proof of address ownership in ARP leads to a range of serious security vulnerabilities. Previous solutions to ARP have failed to simultaneously address the compatibility and cost requirements of current networks.

We have introduced TARP: a Ticket-based Address Resolution Protocol, and detailed its implementation. Built as an extension to ARP, TARP achieves resilience to cache poisoning. We have shown experimentally that TARP reduces cost by as much as two orders of magnitude over existing protocols.

ARP vulnerabilities will remain a serious network security problem until a viable alternative is accepted. We have shown TARP to be viable, but much work remains before our implementation can be broadly used. Acceptance by the Internet community and implementation of TARP not only in popular operating systems but also in network devices is needed to make ARP based attacks a thing of the past.

## References

- [1] Anatomy of an ARP poisoning attack. <http://www.watchguard.com/infocenter/editorial/135324.asp>, accessed June 2005.
- [2] The OpenSSL library. <http://www.openssl.org/>.
- [3] The packet capture library. <http://www.tcpdump.org/>.
- [4] C. Adams and R. Zuccherato. A general, flexible approach to certificate revocation, June 1998. <http://www.entrust.com/securityzone/whitepapers.htm>.
- [5] W. Aiello, J. Ioannidis, and P. McDaniel. Origin Authentication in Interdomain Routing. In *Proceedings of the 10th ACM Conference on Computer and Communications Security*, pages 165–178. ACM, October 2003. Washington, DC.
- [6] R. Arends, R. Austein, M. Larson, D. Massey, and S. Rose. RFC 4033, DNS Security Introduction and Requirements. *Internet Engineering Task Force*, March 2005.
- [7] R. Arends, R. Austein, M. Larson, D. Massey, and S. Rose. RFC 4034, Resource Records for the DNS Security Extensions. *Internet Engineering Task Force*, March 2005.
- [8] R. Arends, R. Austein, M. Larson, D. Massey, and S. Rose. RFC 4035, Protocol Modifications for the DNS Security Extensions. *Internet Engineering Task Force*, March 2005.
- [9] J. Arkko, T. Aura, J. Kempf, V. antyl, a Nikander, and M. Roe. Securing IPv6 neighbor and router discovery. In *the ACM Workshop on Wireless Security*, Sept. 2002.
- [10] J. Arkko, J. Kempf, B. Zill, and P. Nikander. Secure neighbor discovery (send). RFC 3971, Mar. 2005.

- [11] S. M. Bellovin. Security problems in the TCP/IP protocol suite. *Computer Communications Review*, 2(19):32–48, Apr. 1989.
- [12] S. M. Bellovin. A look back at “security problems in the TCP/IP protocol suite”. In *20th Annual Computer Security Application Conference (ACSAC)*, pages 229–249, Dec. 2004.
- [13] D. Bruschi, A. Orgnaghi, and E. Rosti. S-ARP: a secure address resolution protocol. In *Proceedings of the 19th Annual Computer Security Application Conference (ACSAC)*, 2003.
- [14] Cisco Systems. *Catalyst 4500 Series Switch Cisco IOS Software Configuration Guide, 12.1(19)EW*. [http://www.cisco.com/en/US/products/hw/switches/ps4324/products\\_configuration\\_guide\\_chapter09186a008019d0de.html](http://www.cisco.com/en/US/products/hw/switches/ps4324/products_configuration_guide_chapter09186a008019d0de.html), accessed May 2005.
- [15] Cisco Systems. *Configuring DHCP Snooping*. [http://www.cisco.com/univercd/cc/td/doc/product/lan/cat4000/12\\_1\\_13/config/dhcp.pdf](http://www.cisco.com/univercd/cc/td/doc/product/lan/cat4000/12_1_13/config/dhcp.pdf), accessed April 2006.
- [16] Cisco Systems. *Configuring Dynamic ARP Inspection*. <http://www.cisco.com/univercd/cc/td/doc/product/lan/cat6000/122sx/swcg/dynarp.pdf>, accessed April 2006.
- [17] R. Droms. Dynamic host configuration protocol. RFC 2131, Mar. 1997.
- [18] R. Droms and W. Arbaugh. Authentication for DHCP messages. RFC 3118, June 2001. <http://www.ietf.org/rfc/rfc3118.txt?number=3118>.
- [19] C. Ellison, B. Frantz, B. Lampson, R. Rivest, B. Thomas, and T. Ylonen. RFC 2693, spki certificate theory. *Internet Engineering Task Force*, Sept. 1999.
- [20] B. Fleck and J. Dimov. Wireless access points and ARP poisoning: Wireless vulnerabilities that expose the wired network. <http://downloads.securityfocus.com/library/arppoison.pdf>.
- [21] J. Galvin. Public Key Distribution with Secure DNS. In *Proceedings of the 6th USENIX Security Symposium*, pages 161–170, July 1996.
- [22] M. Gouda. and C. Huang. A secure address resolution protocol. *Computer Networks*, 41:860–921, Jan. 2003.
- [23] C. A. Gunter and T. Jim. Generalized certificate revocation. In *POPL '00: Proceedings of the 27th ACM SIGPLAN-SIGACT symposium on Principles of programming languages*, pages 316–329, New York, NY, USA, 2000. ACM Press.
- [24] R. Housley, W. Ford, W. Polk, and D. Solo. RFC 2459, Internet X.509 Public Key Infrastructure Certificate and CRL Profile. *Internet Engineering Task Force*, January 1999.

- [25] J. Kempf, C. Gentry, and A. Silverberg. Securing IPv6 neighbor discovery using address based keys (abks). <http://www.watersprings.org/pub/id/draft-kempf-abk-nd-00.txt>. [draft-kempf-ipng-secure-nd-00.txt](http://www.watersprings.org/pub/id/draft-kempf-ipng-secure-nd-00.txt) work in progress.
- [26] P. Kocher. On Certificate Revocation and Validation. In R. Hirschfeld, editor, *Financial Cryptography FC '98*, volume 1465, pages 172–177, Anguilla, British West Indies, February 1998. Springer.
- [27] L. B. N. L. (LBNL). ARPWatch: Ethernet monitor program. <http://www-nrg.ee.lbl.gov>, accessed May 2005.
- [28] W. Lootah, W. Enck, and P. McDaniel. TARP: Ticket-based address resolution protocol. In *Proceedings of the 21st Annual Computer Security Application Conference (ACSAC)*, 2005.
- [29] M. Malkin, T. D. Wu, and D. Boneh. Experimenting with shared generation of rsa keys. In *Proceedings of Network and Distributed Systems Security 1999*. Internet Society, February 1999. San Diego, CA.
- [30] P. McDaniel and S. Jamin. Windowed Certificate Revocation. In *Proceedings of IEEE INFOCOM 2000*, pages 1406–1414. IEEE, March 2000. Tel Aviv, Israel.
- [31] P. McDaniel and A. Rubin. A Response to ‘Can We Eliminate Certificate Revocation Lists?’. In *Proceedings of Financial Cryptography 2000*. International Financial Cryptography Association (IFCA), February 2000. Anguilla, British West Indies.
- [32] R. Merkle. Secure communications over insecure channels. *Communications of the ACM*, pages 294–299, Apr. 1978.
- [33] S. Micali. Efficient Certificate Revocation. Technical Report Technical Memo MIT/LCS/TM-542b, Massachusetts Institute of Technology, 1996.
- [34] D. L. Mills. RFC 1301, network time protocol (version 3) specification, implementation. *Internet Engineering Task Force*, March 1992.
- [35] M. Myers. Revocation: Options and Challenges. In R. Hirschfeld, editor, *Financial Cryptography FC '98*, volume 1465, pages 165–171, Anguilla, British West Indies, February 1998. Springer.
- [36] M. Myers, R. Ankney, A. Malpani, S. Galperin, and C. Adams. RFC 2560, X.509 Internet Public Key Infrastructure Online Certificate Status Protocol - OCSP. *Internet Engineering Task Force*, June 1999.
- [37] T. Narten, E. Nordmark, and W. Simpson. Neighbor discovery for IP version 6 (IPv6). RFC 2461, Dec. 1998.
- [38] M. Noar and K. Nassim. Certificate Revocation and Certificate Update. In *Proceedings of the 7th USENIX Security Symposium*, pages 217–228, Jan. 1998.
- [39] A. Ornaghi. S-ARP: a secure address resolution protocol. <http://security.dico.unimi.it/research.it.html#sarpd>, accessed May 2005.

- [40] D. C. Plummer. An Ethernet address resolution protocol or converting network protocol addresses to 48.bit Ethernet address for tansmission on Ethernet hardware. RFC 826, Nov. 1982.
- [41] C. Rigney, S. Willens, A. Rubens, and W. Simpson. RFC 2865, remote authentication dial in user service (RADIUS). *Internet Engineering Task Force*, June 2000.
- [42] R. Rivest and B. Lampson. SDSI a simple distributed security infrastructure, Oct. 1996. <http://theory.lcs.mit.edu/~rivest/sdsi11.html>.
- [43] M. Schiffman. The libnet packet construction library. <http://www.packetfactory.net/libnet/>.
- [44] K. Seo, C. Lynn, and S. Kent. Public-Key Infrastructure for the Secure Border Gateway Protocol (S-BGP). In *Proceedings of DARPA Information Survivability Conference and Exposition II*. IEEE, June 2001.
- [45] D. Song. dsniff: a collection of tools for network auditing and penetration testing. <http://www.monkey.org/~dugsong/dsniff>, accessed May 2005.
- [46] M. V. Tripunitara and P. Dutta. A middleware approach to asynchronous and backward compatiable detection and prevention of arp cache poisoning. In *15th Annual Computer Security Application Conference (ACSAC)*, pages 303–309, 1999.
- [47] T. Ylonen. SSH - Secure Login Connections Over the Internet. In *Proceedings of the 6th USENIX UNIX Security Symposium*, pages 37–42. USENIX Association, June 1996. San Jose, CA.