Security Analyses of Network Time Services

Matt Bishop

Department of Mathematics and Computer Science
Dartmouth College
Hanover, NH 03755

ABSTRACT

In this paper we discuss the generic security requirements for a network time service, analyze the nature of the mechanisms necessary to provide these requirements, and apply our analysis to an existing network time protocol. The analysis reveals several shortcomings.

1. Introduction

The accuracy of local clocks on most computer systems is woefully inadequate, errors of minutes being common and of hours, days, or even years occasional. Among the causes are imprecise initialization (due to the inaccuracy of a system administrator's watch), clock drift (increasing imprecision of hardware clocks leading to inexact incrementing of the local time counter), or a failure to observe local or international time conventions (such as leap seconds or a change to Daylight Savings Time). This imprecision can have unpleasant effects on networked systems.

Many operating systems record the time at which each file was last modified. The time of the last backup is also recorded, and only those files which have changed since the last backup are backed up. Suppose backups are made daily, and after one backup the system clock is reset to one day before the time recorded for the just-completed backup. At the next backup, no files will be saved even if they had changed since the last backup.

A second example involving authentication shows how clock imprecision can affect systems using timestamps. The Kerberos authentication system [20] implements the Needham-Schroeder protocol [15] with Denning and Sacco's modification [2], which uses timestamps to solve the replay problem. The system timestamps each authentication message when it is generated at the authentication server, and also imposes a time limit on the validity of that credential. If the authentication server's clock is out of sync with those of the systems which check authentication credentials, the messages will either never be valid or they will be valid for longer than intended. In the former case, the inability to authenticate would inhibit operation of the system, and in the latter case, the extended period of validity would increase the vulnerability of the system to replay attacks (for example).

The goal of a *time distribution protocol* is to deliver continuous, accurate time synchronized with reference standards even when leap seconds occur [13]. Such protocols establish a set of primary time reference sources which are directly synchronized with external sources. These may communicate with secondary servers, which in turn may communicate with other (secondary) servers designed to propagate time to hosts on a subnet; the servers propagate the time either by initiating transmission of time messages or by responding to requests from clients seeking the time.

A time distribution protocol is closely related to a *time synchronization protocol*, the difference being that the latter does not attempt to synchronize time with a reference standard. Rather, it seeks to keep the clocks of several computers within some (small) interval of one another; but this time may be very different than the physical time of day. Time synchronization protocols, as opposed to time distribution protocols, are often used when reference to a national standard is unwieldy, impossible, or unnecessary. Note that a time distribution protocol may be used as a time synchronization protocol once a mechanism to establish the reference time is in place.

The goal of a *network time service* is to allow a system to synchronize its clock with those of known, accurate primary time servers. This means synchronizing time (so the clocks agree on the time of day) and synchronizing frequency (so the clocks appear to tick at the same rate). However, the propagation of time messages over a network is hindered by transmission delays, unreliable connections, disparity of methods of clients obtaining the time, and heterogeneousness of computing resources. These factors should not affect the synchronization of the clocks; a time service must provide accurate time even in the face of large (statistical) delays during propagation, as well as being very redundant, so the loss of a single subnet or transmission path does not prevent other portions of the network from obtaining the correct time. Further, the protocol should be flexible enough to work with a variety of client/server interfaces, such as having clients continuously poll for the time, or obtain it by remote procedure calls, as well as in broadcast, multicast, and point-to-point transmission modes.

Time distribution protocols may be decentralized or centralized. In a decentralized time distribution protocol, time servers synchronize each other. A centralized service has one or more time servers which have the correct time, and other servers are driven from it [13]. Given that the correct time is a national standard, we shall assume a centralized model, with a set of *primary time servers*

Version 1.1 Dated April 7, 1992

time messages, thereby informing that server's clients (and, recursively, all other clients relying on the server) of the incorrect time. The clients would then either accept the incorrect time as correct, or (at the least) discard the server as a valid source. The second attack is more subtle, and effective only if the server expects messages from its clients. In that case, the attacker impersonates the client, and then sends erroneous messages to the server (or simply fails to respond to the server's messages). In that case, the server might conclude the client is down and simply stop sending time messages to it. The effect would be the same as with the first attack; all clients depending on the impersonated client would be threatened.

If an attacker cannot impersonate a server, he may be able to *modify* the time messages coming from the server. This attack is harder to launch, especially if a statistical mechanism is used to filter time messages, as the attacker must modify *all* messages within a certain interval, and can do so only within limits constrained enough so as to prevent rejection by the client. However, if done appropriately, this attack can be even more devastating than a masquerade attack, because it does not require elimination of the network connections of the real server and so is less likely to be noticed.

A third attack is much simpler than the first two, requiring no modification to any network information tables or to packets on the network. The attacker simply records the time messages from a server or a client, and at a later time *replays* them. The threat is that as the time message is genuine (just late), the client might accept it as being current.

This attack is distinct from one involving *delay*; in the latter, time messages are blocked for some period of time. The effect, being received by the client when the information in the time message is no longer current, is the same as for a replay attack; the difference is that replayed messages are seen (at least) twice, whereas delayed messages are seen once. Both pose the same threat – the client may accept them as current, and either reset its clock to an incorrect time or use the incorrect time in an algorithm which statistically "smooths" time from several sources, thereby skewing the client's notion of the correct time.

A well-designed *denial of service* attack could cause a client's notion of time to vary from that of the network time service. In this attack, an attacker deletes time messages, thereby eliminating a (reliable) source for the client; if this can be done with all (reliable) sources upon which the client depends, its clock may drift unacceptably.

---

2. For simplicity, we shall refer to "attackers" using the male pronouns. But some of the best attackers are, and have been, female, so the pronoun should be read in the more general sense of "entity."

Closely related to denial of service and masquerade is an *illegal access* attack, in which a server exceeds its authorizations. Although there are network management protocols extant (such as SNMP), it is quite reasonable to add to a network time service an ability to use special-purpose commands to alter parameters within the client; this would be especially useful for debugging or monitoring the correctness of the assumptions underlying the statistical algorithms used to compensate for network delays. However, if an attacker could gain authority to set the time on a client, he could wreak havoc. Such an attack requires that the time service allow clients to have a set of sources which is neither fixed nor configured at start time.

There are other attacks which are much less relevant to this type of system, but about which a few comments are in order. Many of these attacks arise when a vendor (or other entity) is charging for the use of the network time service, and must be separated from the accounting by which the clients are charged. This is necessary because in many cases use of the network time service involves mutual trust – the vendor trusts the client not to alter his software, and the client trusts the vendor to supply the correct time. This brings up issues of operating system security and trustworthiness of software, and the scope of these issues is well beyond the purview of this paper.

An example of this would be when some way is provided to ensure that if a time server (or client) begins to drift due to erroneous messages from another source, the erroneous source could be tracked down and be unable to deny the error (*non-repudiation of origin*). In the context of maintenance, this would be a useful property. In the context of security, it is not useful. If a (non-temporal) nonce were used to order transmissions, the order could be certified and by comparing the timestamps on these messages, erratic ticking may be revealed. But a good statistical algorithm would either smooth the irregularities or determine they could not be compensated for and discard time messages with (statistically) erroneous timestamps; this could lead to discarding the source as a legitimate, trustworthy clock source. So the only use for non-repudiation would be in conjunction with other services provided by the client host. In other words, the timestamps must be bound to the (internal) service, which enlarges the scope of the non-repudiation aspect beyond that of simple non-repudiation of a time message. So, for our discussion, we shall not consider repudiation of origin to be an attack upon a network time service.

Traffic flow analysis is not specifically related to a network time service, but applies to any network service. In that context, it simply provides a mechanism for identifying clients and sources, which may be useful in deciding which hosts to attack. However, as this technique provides no information about attack mechanisms, we shall not deal with it further.

Routing control is normally under control of the network layer, not the application layer where the time service resides. If an attacker can control the routing, then it could route time messages along routes with long delays, congestion, or other problems. Like traffic flow analysis, this simply enables attacks to be made; it provides no information about attack mechanisms. Further, because the use of routing control to reduce security depends so highly on network topology and levels of trust in both the network itself and the hosts on the network, it requires specific knowledge of the network and routing mechanisms in use. As our context is that of a generic network time service only, we shall therefore not discuss it further.

Non-assurance of time message receipt and denial of receipt deal with preventing the source from being able to prove time messages reached their intended destination, and are most likely relevant only if a client is paying for the use of a network time service. As the client software would not be under the control of the organization providing the time service, but would reside on a machine under the control of the paying organization, the administrator of that client can do whatever he or she wishes. In the absence of tamperproof and certifiably correct software, this problem is at the heart of operating system security and hence we shall not deal with it here.

In a similar vein, if a vendor is selling the use of a network time service, it would be in his best interest to keep those who have not paid for the use of that service from seeing those messages. But other than this specific circumstance, there seems little danger to an attacker simply observing time messages, so we shall assume that *passive wiretapping* (or *eavesdropping*) poses no threat to the correct functioning of a network time service.[3]

Finally, *changing order of transmission* is equivalent to a delay attack, in which the message moved elsewhere in the ordering was simply delayed. Like controlling routing, we shall not consider this problem separately.

Given these attacks, let us now consider the mechanisms that either limit or defeat them. We shall of course only focus on those mechanisms relevant to the attacks we described as threats to the network time service itself.

A masquerade attack can be handled by requiring the party or parties involved to authenticate themselves. This usually involves cryptography, and raises issues of key distribution as well

---

[3] We should note that use of the reference clocks that would synchronize a network time service's primary services do not require paying a fee. The only charges are for the telephone call; and as the reference clocks also broadcast time over radio stations, even that charge can be avoided.

as performance, especially in the context of a time service, where any variable delay can adversely affect the correct operation of the system.

Message modification can be defeated by using a mechanism to ensure the integrity of time messages. Very often this technique is combined with authentication; either the entire message is encrypted in such a way that decrypting a modification will produce clearly erroneous output, or a message integrity check is encrypted. Should the message be modified, the computed message integrity check will not match the transmitted one.

Both replay attacks and delay attacks are difficult to handle when the client has no reliable source. When a time message comes in, it must be assumed accurate in the absence of any contradictory information. Hence, to defeat these attacks, the clients should have multiple sources and employ an algorithm which can compensate for delays and can eliminate obviously incorrect messages. Note that special care must be taken with replayed messages; they cannot simply be discarded, as the earlier message may have been in error and the source resynchronized. Also notice the relationship here between security and reliability; a delay may simply mean the network is congested, not that the time service is under attack.

Denial of service attacks also require a redundancy of sources, the reasoning being that, so long as one source is still available, the client need not rely on its own internal clock. With many sources, an effective denial of service attack would require all sources being inaccessible to the client; eliminating only some still allows the client to synchronize to external sources.

Finally, attacks based on unintended access can be controlled by access controls. There are essentially three kinds of access controls, *mandatory access controls* (sometimes called *rule-based access controls*), *discretionary access controls* (sometimes called *identity-based access controls*) and a third form in which the original owners of data retain control of the data as it propagates throughout a system[4] (we shall refer to this control as *originator-based access controls* or *original access controls*). Mandatory access controls are primarily used to protect classified or proprietary data, and it is difficult to conceive of a situation in which the standard time of day would be considered classified or proprietary. Similarly, original access controls are used when the original owners of data need to maintain tight controls on how the data is disseminated, and it is equally difficult to conceive of a situation in which one would desire, let alone be able to, retain tight controls on

---

[4] This form is relatively new [6][10] and the precise functioning of the mechanism depends on which variant is used. However, all variants enable the original owners to retain some sort of control over the data as is propagates throughout the system.

possible attacks	*a threat to NTS*	*mechanisms for prevention*
masquerade	yes	authentication
modification	yes	integrity checking
replay	yes	use of sequence numbers or nonces
denial of service	yes	redundency of sources
delay	yes	timestamping
unintended access	yes	access controls
non-confidentiality	no	
non-assurance of receipt	no	
traffic flow analysis	no	
routing control	yes	see delay, denial of service
deniability of receipt	no	
repudiation of origin	no	

Table 1. Attacks, NTS vulnerabilities, and Mechanisms for Prevention

the current time of day or date. So these security mechanisms have little use in network time services.

However, discretionary access controls can be used to prevent unauthorized clients (or servers) from setting or obtaining the time. Three modes of access are required: a *get time* mode allows the client to request the server to send a time message; a *set time* mode allows the server to synchronize with the source of a time message, for example if a secondary server drifts, a primary server would use this mode to reset the secondary server; and a *command* mode allows commands other than the above two to be executed, for example to perform maintenance or debugging functions.

Table 1 summarizes the possible threats to a network time service and the use of security services to defeat those attacks, or at least ameliorate their effectiveness. To summarize, a reasonable set of security services would be authentication, integrity checking, identity-based access control, detection of duplicate time messages, replay detection and prevention, and redundancy of sources. As the first two can defeat masquerade and modification attacks, they are absolutely necessary. The third is necessary to enable determination of valid sources if not all sources are to be equally trusted; it also requires authentication so the identity upon which the access control decision is based cannot be spoofed. Detection of duplicate time messages and replay attacks are basically the same thing, and unfortunately cannot be prevented, but when detected, the messages can be discarded and (ultimately) the source(s) involved can be ignored. However, without sufficient redundancy of sources, this could result in isolation of a client, meaning that the client's clock would gradually drift from the correct time.

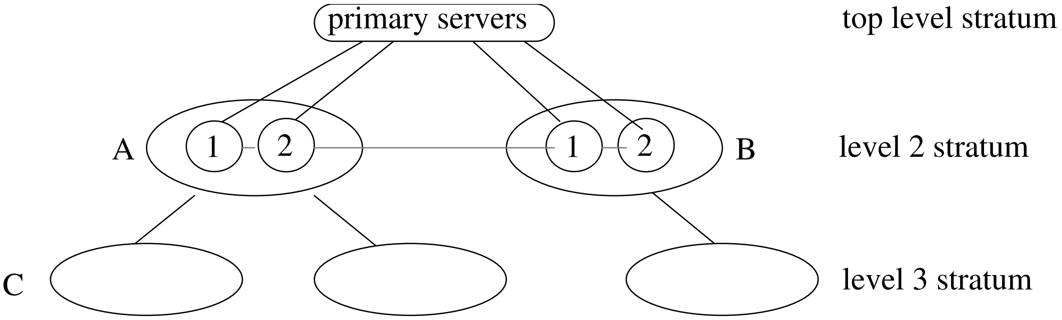


Figure 1. The NTP hierarchy. The ellipses represent sets of cohorts, the solid lines synchronization paths, and the dotted lines paths along which time information is exchanged. A server in C can be synchronized by one in A but not by one in B. If the synchronization path from the primary servers to server 1 of B were to be disrupted, then that server would be synchronized by server 2 in B but not by server 2 in A (as it is in a different cohort). As the shortest path between server 1 in B and the primary servers through its clock source is of length 2, then server 1 of B would be at stratum 3.

The ability of the above services to defeat or inhibit attacks depends greatly upon how they are implemented. We now turn to an existing time distribution protocol and network time service to demonstrate the manner in which the above framework can be used to analyze the security of a time service.

3. Network Time Protocol Version 2

The Network Time Protocol (or NTP) [11][12][13] is a protocol designed to provide network time service in a wide-area network. It designates several sites as *primary time servers*; these communicate with *secondary time servers* over *synchronization paths* which are said to connect *peers*. The secondary time servers also communicate with other secondary time servers; in addition, each such node serves clients on a subnet. The *stratum number* is a measure of distance from a primary time server, specifically the number of synchronization paths that must be traversed to get from the primary time server to the secondary time server. Because network failures must not affect the availability of the time service, the synchronization paths are not fixed, but may be reconfigured as needed.

Primary time servers are synchronized by an external system (such as radio or atomic clocks) with up to 232-picosecond (2×10^{-10} seconds) resolution. Secondary time servers are synchronized by primary time servers or other secondary time servers with lower stratum numbers. The arrangement is hierarchical, with members of a set (called *cohorts*) initially at stratum level i serving some group at level $i+1$ (see Figure 1). Members of the group at $i+1$ may synchronize themselves with any time server in the set, but not with any server not in the set (even if it is at a lower stratum). Membership in a cohort is determined simply by the stratum number of the host and the selection algorithm of the peer that can be synchronized by members of the cohort.

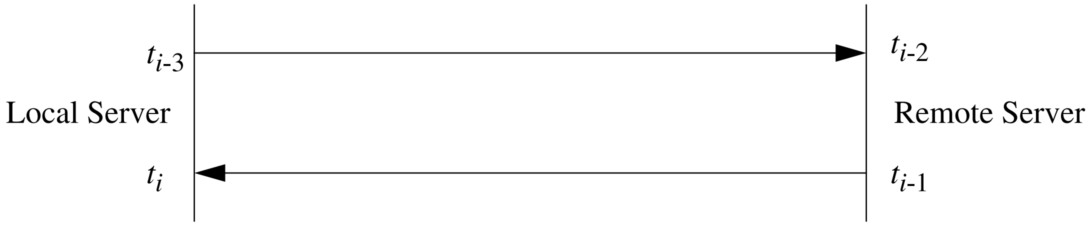
When a message arrives at an NTP time server, it either causes an *association* (instantiation of the protocol machine) to be created, or causes an existing association to act; what happens depends on the mode of the association. The two basic functions are to synchronize another host's clock, or to be synchronized by another host's clock.

3.1. Association Modes

Three operating modes are designed for use on high-speed local area networks, although they may be used on wide area networks as well. The first set allows non-server hosts to synchronize themselves to NTP servers. An association operating in *client* mode periodically sends messages to its peer; an association operating in *server* mode, which is created when a message from another association operating in client mode arrives, replies with the server's idea of the time, and then terminates; and an association operating in *broadcast* mode sends periodic time messages. The client association may resynchronize the host's local clock, but no association in broadcast or server mode will ever reset its host's time.

The primary and secondary time servers rely on two other modes to synchronize themselves, with higher-level servers synchronizing lower-level ones. An association in *symmetric active* mode periodically sends messages intended to synchronize other hosts. When the messages arrive at these peers, an association in *symmetric passive* mode is created. If the source of the message is at a higher stratum than the current host, a reply is sent and the association terminates. Otherwise, the source synchronizes the current host as indicated by the message; the current host responds with a time message of its own. Normally, the servers at the highest strata will run in symmetric active mode, with servers at lower strata in both symmetric active and passive modes. Note that a host may acquire peers either through receipt of messages or through initialization data read at configuration.

In addition to synchronization messages, NTP allows several types of control messages designed to handle exceptional conditions. These messages do not normally cause synchronization, but instead communicate (or set) information related to the current association, or indicate exceptional events. They are designed for use when no other network management facilities (such as SNMP [1]) are available, and these commands may be sent by other than NTP peers. Further, the NTP specification does not require implementations to be able to process these control messages.



clock offset $c_i = ((t_{i-2} - t_{i-3})+(t_{i-1} - t_i))/2$ and roundtrip delay $d_i = (t_i - t_{i-3})+(t_{i-1} - t_{i-2})$.

Figure 2. Computation of clock offset and roundtrip delay.

3.2. Selection of Source Peer and Smoothing of Data

NTP uses various algorithms to filter “bad” timestamps from “good,” the discriminator including (among other things) how much the newer timestamp is at variance with previous ones.

The first algorithm attempts to improve the accuracy of estimated clock offsets and roundtrip delays by eliminating bad data. From each NTP message the roundtrip delay *d* and clock offset *c* are computed (see Figure 2). The values computed from the last eight messages are retained and constitute the *sample*. The first algorithm simply chooses from among these the one with the lowest delay and uses its associated offset as the estimated clock offset. It also computes an estimate of the sample dispersion based on clock offsets in the sample

The second algorithm uses this estimate to determine which peer should be allowed to synchronize the clock. It first sorts all possible clock sources by stratum number and then by dispersion from the root of the synchronization subnet (that is, up to the primary server synchronizing the node). The list is pruned using various sanity checks and other criteria described in [12]. The elements of this list then are scanned repeatedly, and during each scan the clock dispersion relative to each peer is computed, and that peer with the highest dispersion is eliminated. This repeats until there is only one element in the list; that is the required source.

3.3. Receive and Packet Procedures

Whenever a packet is received, either an error or a packet procedure is called. If the modes of the host and the peer are incompatible (for example, both are symmetric passive), the error procedure is called, the packet discarded, and the association deleted (unless it is preconfigured).

Otherwise, the packet procedure checks that the packet is reasonable and if so, resets internal variables, adjusts the local clock if necessary, and possibly select a new peer to be used as the clock source. Figure 3 summarizes the procedure; note that the association’s variables will be updated even if the packet is not used to compute delay or offset with respect to the peer.

Once the packet procedure is finished, the receive procedure resumes. If the peer is in client mode, or the local host is in server or broadcast mode, the peer must be sent the local time, so after the frequency of polling by the peer is updated, a reply is transmitted and the receive procedure ends. Otherwise, if the sanity check is not set, or the peer is in symmetric active mode and the local host in symmetric passive mode, the peer is marked as reachable. In any case the packet is discarded and the receive procedure terminates.

3.4. Transmit Procedure

Associated with each peer is a peer timer which decrements periodically. When that timer is 0, an NTP message is generated and sent to the peer. The transmit timestamp is saved to validate the reply. Next, if the packet procedure has not obtained valid roundtrip delay and clock offset measurements from the peer with in the last two time-out intervals, it updates a sample set of (roundtrip delay, clock offset) pairs with (0,0) to skew the estimated dispersion, and then determines if a new time source should be used. The peer timer is reset to the shorter of the peer-host polling interval and the host-peer polling interval, but not to less than 64 or more than 1024 seconds. This ensures the polling frequency varies only within a specific interval.

Finally, the host-peer polling interval is decreased if the estimated dispersion is larger than 0.5, or increased if it is smaller than 0.5, to balance the need for low dispersion with that for low NTP-related network traffic.

3.5. Security Mechanisms

For reasons discussed earlier, we shall consider only those issues raised by relying on the information transmitted over the network. Each NTP message (also called *packet*) contains time

```
if (packet sent when last received packet sent) or
        (last packet peer received sent at time other than time last packet sent) then
                sanitycheck := true;
update association variables to reflect information in packet
if (peer clock not synchronized or not updated for 1 day) or
        (peer not authenticated correctly) or
                (peer not preconfigured and at a higher stratum) or (sanitycheck) then
                        discard message and exit
if (packet originate timestamp = 0) or (time last message received by peer = 0) then
        exit (* note sanitycheck is not set *)
compute delay, offset, corrections, update local clock
```

Figure 3. Summary of the packet procedure.

and association-related information. Three NTP-provided mechanisms access these for security reasons.

3.5.1. Delay Compensation Mechanisms

The most basic mechanism is one to compensate for network delays; it is used to offset problems from statistical irregularities or problems in network connectivity and congestion which might be maliciously created or natural. The precise function used is detailed in [11]; the aspect relevant to this discussion is that the algorithm calculates both the roundtrip delay and the clock offset relative to the peer, and from these applies a statistical procedure to determine estimates used to update the local clock. We shall assume that given accurate data, the source selection and delay compensation mechanisms work as intended.

3.5.2. Access Control Mechanism

This feature requires that the set of all hosts be partitioned into three subsets: those that are trusted, those that are friendly, and all others. Trusted hosts are allowed to synchronize the local clock; friendly hosts are sent NTP messages and timestamps as appropriate, but may not change the local clock; and messages from hosts in the third subset are ignored. The set of trusted hosts is either preconfigured (at initialization) or configured based upon a trusted ticket service such as Kerberos [20]. The peer's IP address as recorded in the NTP packet is to be used as the address upon which access control is based. The implementation of this feature is not specified, although two are suggested (the first, treating all peers configured in symmetric or client modes as trusted and all others as friendly; the second, masking the internet address and looking up the result and the peer mode in a table to obtain the subset to which the peer belongs). This feature need not be supported for an implementation to conform to the NTP specification.

3.5.3. Authentication Mechanism

A third feature, allowed but not required by the specification, is integral to the packets, and is designed to provide both origin authentication and packet integrity: a cryptographic checksum. A major requirement for this integrity check is that the computation time be predictable, since it must be done after timestamping, and the timestamping must reflect the time needed to compute the checksum. The authentication mechanism described in [11] meets these requirements, and is intended for use only until more general, network-wide, authentication and integrity facilities become available. It uses a cryptographically-based message integrity check; all algorithms and keys are distributed by a mechanism other than NTP, and the keys and algorithms are referenced within the packet by indices.

When a packet is transmitted in authenticated mode, the entire NTP packet except for the authenticator and additional information is checksummed using the active peer's key (if available) or the default key 0 (if not). Note that if the association is symmetric active, client, or broadcast, the key used is that of the local host, whereas if the association is symmetric passive or server, the key used is that of the remote host (or the default key).

When a packet is received, the authentication routine is invoked. If the message contains no authentication information, the authentication and integrity check fails; further, if the peer is not preconfigured (at initialization), authentication for that peer is disabled. In either case the routine exits. However, if the message does contain authentication information, the index number of the peer's key is reset to that in the packet, and the checksum is recomputed and compared to the transmitted checksum. If the key is not the default one, and the checksums match, the authentication and integrity check succeeds; otherwise, it fails.

The authentication and integrity mechanism in [11] uses the Data Encryption Standard based DEA-1 algorithm (that is, DES in CBC mode) [3] to compute the checksum. The checksum is 64 bits long, and the key index is 32 bits; these 96 bits are appended to the original NTP packet, and their presence is indicated by the length of the packet. (The choice of algorithm is not part of the NTP specification of the authentication and integrity mechanism.)

We should note that for control messages, if the received message is authenticated, the reply is too. If the received message's checksum is correct, the reply is authenticated using the same key; if not, the reply is authenticated using the default key.

4. Analysis of NTP Security Mechanisms

Version 2 of NTP provides two basic security mechanisms: an access control mechanism and an authentication mechanism. Because they are intended to hinder attacks, we discuss them before turning to the specific attacks that may be launched against an NTP host.

4.1. Access Control Mechanism

The access control mechanism conditions access on the internet address in the source field of the packet. If the attacker can generate or modify an NTP packet, the attacker can choose a source address that allows synchronization of the victim. Hence the access control mechanism is redundant from the point of view of network security; the protection it provides is against a compromised time source, which can simply be denied access (or at least, the ability to synchronize the host).

The problem with the recommended access control mechanism is that, in the absence of an integrity checking mechanism, it relies completely on the unauthenticated source address. As an additional measure, access control can be conditioned on a routing basis; that is, a list of all the intermediate nodes the message passes through is built using the IP record route option [5] (which causes intermediate nodes to insert their address into the IP datagram) and determine access based on that list. This allows an NTP packet from a trusted source to be rejected if it passed through untrusted hosts.

Of course, as the added routing information is not checksummed cryptographically, it can be altered in transit without detection, just as the source address can. For this reason, such a method should be avoided unless the route lies on a trusted network, and gateways to the network will reject packets entering the network with source addresses of hosts on that network. Access control may then be based on that information.

4.2. Authentication Mechanism

First, it should be noted that the authentication mechanism is also an integrity mechanism because it guards against the altering of messages while in transit. The authentication provided is simply that of only two parties (the peer and the host) sharing a common key. In particular, if any *n* hosts have the same key, then it will not be possible to determine which of them sent the message.

Secondly, note that the authenticator is excluded from the integrity checking. This means that the key index associated with the generated cryptographic checksum can be altered without detection. In general, this will cause the checksum to be invalidated and the packet to fail to authenticate correctly; however, if a less robust cryptosystem were used to generate checksums, the authentication mechanism might not prevent undetectable modification of NTP messages.

The integrity checking algorithm suggested in [11] is subject to various cryptanalytic attacks which have been discussed in the literature [7] because the check is only 64 bits long; however, as these attacks are probabilistic in nature, changing the keys periodically will defeat them. However, no key distribution mechanism is defined.

Although we are assuming no system is penetrated, it is worth noting that the keys are used on a per-host basis, not a per-path basis, so compromise of one host's key can lead to compromise of all the hosts it synchronizes.

We should note that the use of a default key is *not* a weakness, contrary to what it would seem. If the checksum is computed using the default key, the flag indicating whether or not the packet is authentic is set to 0 (meaning the packet's integrity or origin are suspect). Hence anything authenticated with the default key will be treated as bogus by the other end.

In what follows we shall assume the authentication mechanism is not compromised.

5. Analysis of NTP with Respect to Attacks

This section describes the attacks that might be launched against an NTP server or client, and how NTP handles them.

5.1. Masquerade

The goal of a masquerade attack is to persuade a timekeeper that the attacker is a peer authorized to synchronize the timekeeper. Note this includes NTP client processes as well as secondary servers. Such an attack involves an attacker sending packets to the victim with source address of the time server to be imitated. As both source *and destination* Internet addresses and ports are matched to find the correct peer, an equivalent attack is simply to change the destination address within the NTP message.

If the host being impersonated is known to the victim and allowed to synchronize the victim, in the absence of access control and authentication, the masquerade may be ignored (but not detected) by the sample processing and selection operations. However, if the attacker alters the timestamps to change the clock offsets and roundtrip delays gradually, those algorithms will provide no protection and the victim's clock will drift from that of the time source.

If the host being impersonated is not known to the victim, and the default is to allow non-preconfigured peers to become the clock source, sending messages in such a way that the victim receives at least 8 messages uninterrupted by any other time source could compromise the time server; since the clock filtering mechanisms use the last 8 messages as the sample upon which outliers are discarded, the attacker needs to ensure it controls the elements of the sample. An even simpler method would be to send messages claiming a very low stratum number, as the selection algorithm would tend to make such a host the synchronization source. For this reason, no non-preconfigured peer should be allowed to become the clock source.

Note that although the timestamps are precise to 80 nanoseconds (and hence it is unlikely the attacker can predict the value of the next time stamp), if the attacker can see the transmitted time of any packet sent from the host to the peer and transmit a (bogus) packet to the host *before* the peer does so, the masquerade will not be detected; but if the host replies, and the reply arrives after the true peer sends another message, the true peer's message will be rejected as bogus because the *pkt.org* field in that packet will not match the time the host sent its last message to the peer. In this way, the attacker could successfully spoof the peer.

The use of authentication would preclude this attack. The use of access control does not; however, it does allow one to "turn off" permission for a suspect server to synchronize the local clock. If access control is used, all non-preconfigured peers should be considered "friendly" at best (using the language of §3.5.2.)

5.2. NTP Message Modification

The goal of a message modification attack is to cause the recipient to incorrectly resynchronize itself, or to disable an active association, by altering a message from one timekeeper to another. By examining the packet procedure, it is clear that several variables related to the association may be changed to reflect information in a packet altered in transit *before* the packet alteration is acted upon. How would such alteration affect the integrity of the recipient's clock?

First, if any of the sanity checks discussed in §3.3. fail, the packet is discarded, the association deactivated (if the message is from a peer that has not been preconfigured), and the clock is not updated. If all the sanity checks are passed, then the clock may be reset (if the strata numbers are correctly related and any access control mechanism indicates the peer is trusted). If not, no harm is done. If the clock is reset, then the fields that the attacker can alter affecting the new time are the timestamp of when the packet left the sender (transmit timestamp), when the peer last received a packet from the local host (receive timestamp), when the peer last sent a packet (originate timestamp) and the precision of the peer's clock. However, for the sanity checks to pass, the originate timestamp must match the time the last packet was transmitted, which is stored in the receiving host, so altering that field will cause the packet to be dropped. Hence only the receive and transmit timestamps, or the precision, could successfully be altered in an attack to change the local system's time.

Altering the precision may result in changes to the roundtrip delay for the packet on systems involving high-speed local area networks. In these cases, a "fudge factor" $\delta = O(2^{precision})$, computed using a system-dependent constant and the precision, is added to the roundtrip delay to

victim's mode	*altered mode*
symmetric passive	symmetric passive, server, broadcast
client	client
server	symmetric passive, server, broadcast
broadcast	symmetric passive, server, broadcast

Table 2. Mode combinations deactivating non-preconfigured associations

account for possible discrepancies between the host and peer clocks. Clearly, by modifying this field appropriately the roundtrip delay can be made (almost arbitrarily) large. It can also be made quite small by choosing a value so that δ is approximately 0; this has as an advantage that the clock source selection algorithm bases its choice of peer in part upon δ.

We should note that the roundtrip delay and clock offset are used to compute both an adjustment to the frequency with which messages are sent to the peer on the local host's initiative (that is, not in response to a peer's message) and to determine which of the set of possible clock sources should be used as the source. Both these computations involve a statistical (weighted) average of the peer clock offsets as well as estimates of the roundtrip delays and clock offsets. Hence altering receive and transmit timestamps, or the precision can also affect the choice of clock source and the frequency of initiating contact with other time servers.

If the purpose of the attack is something other than incorrectly altering the victim's local clock, a variety of other mechanisms may be used. For example a denial of service attack may be launched in a number of ways:

- The attacker could alter the version field to reflect an earlier version of NTP will cause the packet to be discarded unless specific exception has been made. If such exceptions have been made, the result of the attack would depend upon the actions they dictate.
- If the attacker alters the mode field, and the association is not preconfigured, the association may be disconnected if the mode field and the mode of the victim so dictate (see Table 2). Note that under no circumstances can a change of packet mode force a clock update unless the packet would have caused one anyway. Thus, altering the mode can only cause a denial of service.
- Altering the reference timestamp field (which holds the time when the peer's clock was last updated) can cause the association to be discontinued. If it is over one day different than the packet's transmit timestamp field, the packet will be discarded and (if the association is not preconfigured) it will be discontinued. In any case, the state variables associated with the association are updated to those of the packet.

In summary, denial of service attacks are simple to launch and are very effective.

Related to denial of service attacks are attacks which increase the network traffic; these can be used to increase delay time. The frequency of polling clock sources can be affected in two ways:

- The attacker can modify the peer poll interval field of the packet, which contains the minimum interval between messages sent by the peer. The victim then resets the time to initiate a message to the peer to a function of the smaller of its polling interval and the peer's polling interval. However, if the value falls outside a configuration-defined interval, the value of the appropriate bound is used. Hence one can affect the polling interval, but only within specified limits. So one cannot arbitrarily increase the network traffic using this attack.
- Altering the synchronizing distance field alters the victim's perception of the estimated round-trip delay, which affects the frequency of polling the peer. Again, changing this can only alter the polling frequency within specific bounds.

Thus the threat of flooding a network with time messages is minimal.

A more subtle attack would involve forcing the victim to use an unreliable, or attacker-controlled, system as a clock source. The selection of clock source can be influenced in a number of ways:

- If the attacker changes the packet's stratum field from a value greater than the victim's stratum number to a value less than the victim's stratum number, the altered value will replace the original value in the victim's table of peer associations; this peer then becomes eligible to be added to the list of clock sources. Note that access control mechanisms may prevent this if the peer whose packets are being modified is not trusted, or the list of clock sources is preconfigured.
- Altering either the synchronizing distance or dispersion fields alters the victim's perception of the estimated round-trip delay and dispersion of the peer to the primary source, which affects the selection of a clock source. Similarly, altering the packet time reference source alters the victim's perception of the source of its peer; this does not affect the time service, but may affect operator actions.

It is not possible to prevent message modification from affecting the local host time as the distance and dispersion fields of packets from legitimate clock sources can be modified. Further, given that association variables are reset even if the packet is determined to be corrupt or otherwise unusable, the message modification attack allows all aspects of the association to be tampered with. Denial of service attacks are also possible, as is increasing the amount of traffic between peers. Fortunately, the NTP protocol provides an authentication and integrity-checking mechanism which will detect modifications to messages assuming the keys associated with the message are known only to authorized parties.

5.3. Replay

The goal of a replay attack is to intercept NTP messages from one timekeeper to another and retransmit them to cause the recipient to incorrectly resynchronize itself, or to disable an active association.

First, note that the sanity checks will detect replay attacks if the victim sends any message to the originator of the packet, because one of the sanity checks compares the time the victim sent a last message to the peer with the time the peer last received a message from the victim. Hence for a replay to be effective, either the packet being replayed must arrive at the victim before it sends the peer anything, or the packet must be altered to contain this time. The latter essentially implies a passive wiretapping to monitor packets from the victim to the peer (given the resolution of the timestamp and the unlikelihood of the attacker's predicting it exactly).

In the former case, it is not possible to flood the victim in order to force all elements of the sample to reflect the replayed time, because the first step of the packet procedure determines if this packet is the same as the one received previously from the peer. If so, it is discarded. Hence the replayed data will mingle with valid (new) data, and the clock filtering and selection algorithms will cause that data to be ignored if the other packets from the peer are accurate. This suggests one possible attack: record two packets from the peer (being careful that nothing is sent from the victim to the peer in that interval), and then rapidly replay the packets, alternating them. This will make four sample elements be the first packet and the other four the second. However, as the packets are received, the clock offsets and delays computed from the timestamps will become greater and greater, resulting in the peer's estimated dispersion and delay increasing; the clock selection algorithm will simply drop the peer as a valid source (if there are multiple clock sources, the peer will be an outlier; if the peer is the only source, it will at some point have too large an estimated delay). Hence replay will either have a negligible effect, or isolate the victim, and therefore is a denial-of-service attack.

If the delay is greater than the polling interval or no other message has been sent from the peer to the target, the sanity checks in the receive procedure will detect the replay; but as noted in the previous section, this can still cause various association parameters to be reset. In particular, if the synchronization paths have been reconfigured so the peer's stratum number has dropped (and hence the target's stratum number has dropped), the peer could become a source. Otherwise the effects are the same as for message modification.

A major effect of a replay attack will be to reset the recipient's clock backwards; as the message is valid but for an earlier time, if the replay is not caught and the victim resynchronizes its clock to the (replayed) time in the packet, the local time will be reset to an earlier time.

Decreasing the bounds of the polling interval will decrease the window of vulnerability. Another way, involving some protocol redesign, is to change the first sanity check in the packet procedure to reject any message with a transmit timestamp older than the last one received, and create a special resynchronize message to be sent when a clock is changed backwards. Then the window of vulnerability exists only when a resynchronization packet is sent.

If authentication is used, a less suitable alternative is to employ route-based access control as described in §4.1. If a packet comes over an untrusted portion of the network and contains a time that is earlier than the current time, it is rejected as a possible replay attempt. It should be noted that the recorded IP routing information is not cryptographically checksummed, so it is not reliable; but if the final portion of the route is over a trusted portion of the network and gateways to the network will reject packets entering the network with source addresses of hosts on that network, then it may be possible to determine that the packet entered that part of the network from an untrusted part.

5.4. Delay

An alternate way for an attacker to cause a timekeeper to incorrectly resynchronize itself, or to disable an active association, is to delay NTP messages from a clock source. As the delay compensation and source selection algorithms are statistical in nature, they would seem to be vulnerable to this attack. In fact, delaying NTP messages does increase the estimate of delay to the peer; if more than 8 packets are so delayed (so the estimate of the delay is more than 8 seconds), the peer whose packets are being delayed cannot be a source. This may result in the target having no source, resulting in a denial of service attack.

The only way to prevent this is to provide several clock sources, so that if one is deemed unreliable, another source will be used. NTP currently provides this capability.

5.5. Denial of Service

Yet another way to prevent a target from obtaining the correct time is to prevent another timekeeper's NTP messages from arriving at the target of the attack. This could affect the reliability of the time, especially if all sources are blocked; in that case, the site must rely on its local clock and possibly get far out of synchronization with the rest of the Internet.

Like so many denial of service attacks, this one can only be prevented by redundancy of the threatened resource. If a site has several clock sources, one or more of which is deemed unreliable, the clock source selection algorithm will still be able to choose a reliable source for the timekeeper to synchronize with. NTP currently provides this capability.

5.6. Combined Attacks

A combination of the above actions can also prove quite effective during an attack on an NTP server, especially any other attack combined with a denial of service or a delay attack; such combinations attempt to eliminate or hinder communications between a server and members of its cohort not under the attacker's control. For example, denying service to a secondary server from all but one source, and delaying packets from that source, can cause the victim to drift. So can replaying an alternating pair of packets to a server with but one source; the server's time will oscillate between the two values, and the server will report incorrect times. Such attacks can be best dealt with by dealing with each of the component attacks separately.

6. Countermeasures

There are two ways for security mechanisms in NTP to evolve. The first is external to NTP, the second internal. External mechanisms are provided by the network protocols upon which NTP is built; internal protocols assume no underlying security mechanism and implement all such considerations within the NTP protocol. As we shall discuss external measures later, let us consider the internal measures first.

Authentication is a security service basic to the integrity of the timekeeper, and so should always be used; moreover, the computation of the integrity checksum should include the key index (otherwise the integrity of the authenticator is itself suspect). To be most effective, keys should be issued on a per-path, not a per-host, basis. Although the complexity of assigning a distinct key to every peer path used by a server would be high, such a key assignment system adds a fire wall in that if the key for one peer path is compromised, no other peer paths are affected. Further, the different keys do not affect the time needed for authentication, but merely the time needed to administrate the key distribution. As key distribution is out of the scope of the NTP protocol, we merely note that a certificate-based mechanism as used in [8] could be used to distribute keys on a per-peer path basis. There would be a considerable lag involved in validating the keys, but as noted in [11], “the nature of NTP is quite tolerant to such disruptions [as inconsistent key information while rekeying is in progress], so no particular provisions are needed to deal with them.”

Note that the protocol used to distribute keys must not rely on NTP for determining the time during which those keys will be valid, because then if the relevant NTP server were tricked into incorrectly setting its clock, all keys would have invalid periods attached, enabling the attacker to force NTP servers to use current keys (thereby disabling periodic key changes, or the changing of compromised keys). The Kerberos protocol suffers from this problem, as key validity (or the lifetime of the relevant ticket) is determined by a timestamp obtained from the Kerberos server which, presumably, would be set using an NTP server [20]. The certificate-based mechanism mentioned above may have its interval of validity set either automatically or manually, and in any case will be examined by the human responsible for the NTP server when it is issued. Hence it does not suffer from this problem.

The IP record route option can be used as an additional source of information for access control; if the route shows the packet travelled through an untrusted network, one can reject it even if the source is trusted (on the theory that if it really came from the source, it would not have travelled that route). Of course this does not prevent altering the route while the datagram is in transit or at an intermediate node, but it is another detail an attacker will have to worry about.

Note that the authenticator itself is a message integrity checksum. To prevent corrupted packets from affecting the integrity of the timekeeper, the peer association variables should be changed only *after* the packet has passed all sanity checks, especially after the authentication check. Otherwise there is a chance the packet is bogus or corrupt, and in either case the information in it is not reliable and should not be used.

In high-speed LANS, the clock precision field is used to modify the roundtrip delay. Currently this value may range from -127 to 127 inclusive; but it is unlikely that any clock will have precision as coarse as 2^{127} seconds (roughly 5×10^{31} years) or as fine as 2^{-127} seconds (roughly 6×10^{-39} seconds) in the immediate future. To prevent problems from packets in which authentication (and hence integrity checking) is not done, the legal values of this field should be constrained more tightly.

Currently, eight data points are sampled to estimate the dispersion of the clock offset and the roundtrip delay. Because this is a statistical calculation, an attacker can corrupt the integrity of such estimates by flooding the victim with bogus packets. If the sample size can be increased to require more data points, this danger can be diminished. Unfortunately, more than eight points can diminish the stability of the local clock and so diminish the effectiveness of the algorithm [14]. Perhaps allowing some maximum number of packets per polling interval would have the desired effect without affecting the statistics adversely.

The danger of replay arises from the possibility of a system's clock being set *backwards* by a packet from another host. One way to prevent this is to require a special packet be sent when the clock is to be moved back, and provide a nonce to ensure the packet cannot be replayed. (Note it is not sufficient to reject any packet with a timestamp no newer than the last one received, because a clock may run fast and need to be set back; it must then propagate its change to those for which it is the source.) An alternative is to adjust the frequency of the erroneous system clock to slow it down until it shows the correct time; this eliminates the need to reset the clock backwards. However, care must be taken to keep the frequency from being set too low or too high, and any clients of that system must accept the incorrect time until the drift is corrected.

The threat of a denial of service attack can be ameliorated by building into each client a long-term bias that can be used to correct the hardware clock. So, for example, if a client is running for several days and is then isolated, it can use the bias to correct for inaccuracies it has detected in the hardware clock.

Finally, redundancy must be ensured; in particular, no server should have as its source only one other server. NTP does this to a large extent already, but it is imperative that the sets at the various strata contain more than one element. This will limit the effectiveness of delay and denial-of-service attacks.

7. Applicability of External Mechanisms

Rather than require the time service to supply its own security mechanisms, a tempting alternative is to use security protocols provided by the underlying network, and ignore security issues at the higher applications (time service) layer.

There are two problems with such a design. The first is that none of the major security-oriented protocols allow broadcast, because broadcasting unforgeable, authenticated packets would imply the use of a public-key checksumming scheme, and no such scheme runs quickly enough to be used in that context (the best-studied, RSA, runs at 1150 bits/second on a Sun 3/60 [9]; given that the checksum should be on the order of 512 bits, this would mean that at most only 2 packets could be processed per second).[5] The second is that few such protocols are in widespread use.

The lack of broadcast is not serious between primary and secondary, or secondary and secondary, servers, as these are not expected to use broadcast mode; however, for a secondary server providing time service to other hosts on a LAN, the broadcast mode is used. One alternative would simply be to eliminate that mode of operation, and require workstations on such a LAN to query the secondary server directly (the address being configured at boot time). A second would be to allow broadcast but require confirmation by the host being resynchronized.

Unfortunately, the unavailability of such network-level and transport-level protocols is more serious. The current UDP protocol [16] provides for no security beyond that available with IP. The IP options include two relevant here: security and strict source routing.

Strict source routing forces packets to be routed through specific intermediate hosts. If those hosts and the links connecting them are trusted, then packets sent by the time service can also be trusted. However, in a wide-area environment, such assurances are rare; and the source route is specified as a set of fields within the IP datagram itself. Those fields have no associated manipulation detection code. Hence if any link is vulnerable to an active wiretapper, the source route can be altered and the packet made to go along any route.

The IP security option [17] is designed for the protection of information falling under the U.S. classification scheme (*i.e.*, Top Secret, Secret, Confidential, and Unclassified) and is not appropriate for use here.

So, at this point we must conclude that IP does not provide sufficient underlying security to enable its use as an external security mechanism even if broadcasting is eliminated or designated "not trustworthy."

Other protocols not currently in widespread use may prove more suitable. For example, the SDNS Security Protocols SP/3 [18] and SP/4 [19] provide integrity and authentication; this would require time services to detect only replay or delaying attacks. Depending on the design of the time service, though, these may be fundamental to the nature of the time service protocol (for example, as NTP uses a connectionless protocol, it cannot rely on the lower network layers to detect replay)

---

[5] Note that this is not a barrier to NTP, since polling is done no more frequently than on the order of a minute [12].

and so most likely cannot be prevented by the underlying protocol.

8. Conclusion

We have presented a simple model for analyzing the security of network time services, and used it to study the level of security provided by the Internet time service NTP, version 2. Unfortunately, other network time protocols were not designed with security in mind; hence, there are very few protocols against which this model can be tested. It has highlighted the weaknesses of NTP very well, and the list of required (and recommended) security services also suggest remedies.

It is a credit to the designer of NTP that the most crucial threats, those of violation of integrity and failure to authenticate origin of messages, can be countered by conscientious use of the authentication mechanism designed into NTP. With one minor modification (authenticating the key index as part of the message), this reduces the threats to attacks against the statistical algorithms used to derive the correct time. As there is little if anything that can be done about these attacks, short of providing a multiplicity of clock sources, careful analysis of the algorithms provides the best insight into the effectiveness of these attacks. Such studies have been done [13].

This last comment points out the interrelationship of security and reliability. In some sense, this paper has been a study of the reliability of the non-statistical components of NTP. In fact, many of the attacks described in this paper can be "launched" by unreliable network components failing intermittently. Failure can cause either a paucity of information (if links break or sources crash) or incorrect information (because a noisy link corrupts the packets sent over it). Either of these may be caused by an attacker as well.

An analysis similar to the preceding one can be done for time synchronization protocols such as TEMPO [4]. Although they a assume that no clock on the network has the correct time (for example, the master clock of TEMPO simply computes the average of all non-faulty clocks on the network, and then updates them and itself), they do synchronize clocks by changing them to some common time. This common time serves as the "reference standard" for these clocks (although this "reference time" may bear little to no relationship to the real time, and may even be completely inconsistent from one computation to the next).

Security considerations will dictate certain aspects of the design and implementation of future time services. The model presented in section 2 provides a framework for analysis of such time services, and enable designers to anticipate problems and place mechanisms in their protocols to counter these threats.

Acknowledgments. The research leading to the analysis of NTP version 2 was performed at the request of the author of that protocol, Dave Mills, and in conjunction with the Privacy and Security Research Group; as that group has not yet formally reviewed the results, the author takes full credit or blame for it. The framework for analysis presented in section 2 was inspired by work currently being done by the PSRG. Thanks to David Balenson, Stephen Kent, John Linn, Ralph Merkle, Dan Nessett, Mike Padlipsky, Michael Roe, Ken Rossen, Rob Shirey, Miles Smid, Dave Solo, and Steve Wilbur. Special thanks to Dave Mills for his patient explanations of NTP's intricacies, his willingness to listen to my comments, questions, and analyses, and his unfailing good humor.

References

[1] J. Case, M. Fedor, M. Schoffstall, and C. Davin, *Simple Network Management Protocol (SNMP)*, RFC 1157 (May 1990).

[2] D. Denning and G. Sacco, "Timestamps in Key Distribution Protocols," *CACM* **24**(8) pp. 533-536 (Aug. 1981).

[3] Federal Information Processing Standards Publication 81, *DES Modes of Operation* (Dec. 1980).

[4] R. Gusella and S. Zatti, "TEMPO: A Network Time Controller for a Distributed Berkeley UNIX System," *Proceedings of the Summer USENIX Conference* pp. 78-85 (June 1984).

[5] Information Sciences Institute, *Internet Protocol*, RFC 791 (Sep. 1981).

[6] R. Graubart, "On the Need for a Third Form of Access Control," *Proceedings of the Tewlfth National Computer Security Conference* pp. 296-303 (Oct. 1989).

[7] B. Kaliski, Jr., R. Rivest, and A. Sherman, "Is the Data Encryption Standard a Group? (Results of Cycling Experiments on DES)," *Journal of Cryptography* **1**(1) pp. 3-36 (1988).

[8] S. Kent and J. Linn, *Privacy Enhancement for Internet Electronic Mail: Part II – Certificate-Based Key Management*, RFC 1114 (Aug. 1989).

[9] D. Laurichesse, *Mise En Œuvre Optimisee du Chiffre RSA*, Technical Report LAAS-90052, Laboratoire d'Automatique et d'Analyse des Systemes (Mar. 1990)

[10] C. McCollum, J. Messing, and L. Notargiacomo, "Beyond the Pale of MAC and DAC – Defining New Forms of Access Control," *Proceedings of the 1990 Symposium on Research in Security and Privacy* pp. 190-200 (May 1990).

[11] D. Mills, *Network Time Protocol (Version 2) Specification and Implementation*, RFC 1119 (Sep. 1989).

[12] D. Mills, *Internet Time Synchronization: the Network Time Protocol*, RFC 1129 (Oct. 1989).

[13] D. Mills, "On the Accuracy and Stability of Clocks Synchronized by the Network Time Protocol in the Internet System," *ACM Computer Communications Review* (Jan. 1990).

[14] D. Mills, *personal communication* (July 1990).

[15] R. Needham and M. Schroeder, "Using Encryption for Authentication in Large Networks of Computers," *CACM* **21**(12) pp. 993-999 (Dec. 1978).

[16] J. Postel, *User Datagram Protocol*, RFC 768 (Aug. 1980).

[17] M. St. Johns, *Draft Revised IP Security Option*, RFC 1038 (Jan. 1988).

[18] SDNS Protocol and Signaling Working Group, *Security Protocol 3 (SP3)*, Revision 1.5, SDN.301 (May 1989).

[19] SDNS Protocol and Signaling Working Group, *Security Protocol 4 (SP*4), Revision 1.3, SDN.401 (May 1989).

[20] J. Steiner, C. Neuman, and J. Schiller, "Kerberos: An Authentication Service for Open Network Systems," *USENIX Proceedings*, pp. 191-202 (Winter 1988).