AUTHENTICATING TIME AND FREQUENCY SIGNALS

Judah Levine, Time and Frequency Division and JILA,
NIST and University of Colorado

M/S 847, 325 Broadway, Boulder, Colorado, 80303, USA

ABSTRACT

I describe a protocol that can be used to authenticate digital time signals transmitted between time servers and client systems. This authentication is useful to protect the data from being modified en route and to guarantee that the messages really originated from a bona fide server operated by a national timing center. The protocol is based on the standard Network Time Protocol, which is widely used for synchronizing computers on the Internet. The authenticated version of the protocol can be realized using either packet-switched networks (such as the Internet) or standard dial-up telephone circuits. The implementation using dial-up telephone circuits can provide authenticated messages with an accuracy of about 1 ms, but this version requires dedicated modems and telephone lines for its implementation. The Internet version can provide equivalent accuracy in principle, but is vulnerable to a number of attacks that can degrade the accuracy and that are difficult to detect. Either method is suitable for applications that only require a timing accuracy on the order of 1 s. The protocol will be generally available, but will be most useful for systems that are involved in time-sensitive commercial or financial transactions.

1. INTRODUCTION

Packet-switched networks (such as the Internet) are often used to transmit time messages, which can be used to control remote real-time processes or to synchronize the clocks in networked computers. The accuracy that can be realized in this way is only on the order of milliseconds, but this is adequate for many applications. Furthermore, the jitter in the latency of the graphical user interface and the instability in the frequency of the clock oscillator that are used in many computers make it impractical to exploit higher-accuracy messages, even if that class of messages could be transmitted over the network.

The data and the supervisory messages that support the synchronization infrastructure are generally transmitted with only moderate security and cursory authentication. This openness is due in large part to the "sociology" of the Internet, which is based on simple trust models. These models assume that protection and authentication are needed more to protect against programming errors or hardware failures than against malicious attacks.

This openness has served us well until now, and it may continue to be adequate for many (or even for most) network-based applications. However, it is likely to be inadequate for network-based commercial and financial transactions that use the Internet for receiving time-stamps. Protecting these transactions against network-based attacks is a general problem, which must be addressed in the overall design of the network transmission protocols. In this paper I will focus on the special needs and unique requirements associated with using packet-switched networks for authenticated time synchronization. These requirements may make it impractical to use some of the general solutions that have been proposed for protecting the network as a whole because these solutions often do not support methods for obtaining robust estimates of the time delay in sending a message through the network. The resulting uncertainty in the network delay sets a fundamental limit to the accuracy that can be delivered using any synchronization protocol.

The primary applications for these techniques are likely to be time-sensitive commercial and financial transactions where the value of a single transaction is large enough to attract attackers and where something more than the standard, relatively open methods of authentication and security will therefore be required. The number of such applications is relatively small at the moment, but we expect that it will grow rapidly in the near future.

A second important application for these techniques will be to protect the messages that are transmitted between the time servers themselves. A synchronization hierarchy is usually realized using a relatively small number of primary "stratum-1" machines that are connected directly to a primary time reference, and a much larger number of higher-stratum systems, which are synchronized by these primary servers and which in turn synchronize end-node client systems. Insuring the integrity of these messages is very important, since a successful attack at this level can have widespread effects.

2. AUTHENTICATION AND THE NETWORK TIME PROTOCOL

The Network Time Protocol (NTP) [1] is very widely used for transmitting time messages over the

Internet and for synchronizing networked computers. In one common implementation, a client machine periodically queries a server and uses the responses to synchronize its local clock. This protocol supports optional authentication and validation using single-key encryption. When this mode is enabled, the originator of an NTP message computes an authenticator derived from the message using an algorithm driven by a secret key and appends the authentication string to the packet before it is transmitted. The receiver repeats the process of computing the authenticator and compares the value it has computed with the value that it received at the end of the packet. The message is accepted if the two authenticators are equal. If the receiver is a time server, then it usually does not actually use the data in the message. It simply adds time-stamps derived from its local clock, computes a new authenticator and returns the modified message to the originator. (The server may or may not respond to a request when the authentication check on the incoming packet fails. The server usually responds to any request, unless access controls are in use, in which case a failure of this authentication check is taken to imply a violation of the access control.) The originator validates the reply as described above, and, if the message is validated, the data in the reply are used to synchronize its local clock, to apply a time-stamp to an event or for some similar task.

Since the status of the server is not modified by this exchange, we need only focus on how various kinds of attacks might affect the originating client system. In normal operation, the exchange of messages described above is always initiated by the client, and the original outgoing packet is uniquely identified by the time-stamp derived from the clock on the client and inserted into the message when it was first transmitted. Therefore, a simple "replay" attack – in which an attacker tries to fool the client by responding to its current query with an older valid packet is easily detected, since the originating time-stamp in this packet will not match the value that the client expects (assuming that the client software performs this check). The two other possible attacks involve either spoofing the server or delaying the response.

Apart from the relatively weak authentication provided by the Internet protocol suite itself, the ability of a rogue machine to masquerade as a legitimate time server depends on the security of the authentication process. This security, in turn, depends on two factors: the strength of the algorithm that is used to compute the authenticator (which is usually based on a published and well-known method) and the secret key that is used by the algorithm. We assume that the secret key is known only to the legitimate parties to the transaction, and that appropriate measures are used to guarantee the physical security of the keys.

A strong authentication algorithm must satisfy two requirements. First, even though the basic design of the algorithm is known, it should not be possible to recover the key using an algorithm that operates on the message and the authenticator. Second, there should be a negligibly small probability of finding a different message that has the same authenticator. Since the size of the message can be much larger than the size of the authenticator, there are a very large number of individual messages that have the same authenticator. The second requirement is usually satisfied by ensuring that the messages which have the same authenticator are a very small fraction of the total number of messages that might be sent and that there is no algorithmic way of finding them that is more efficient than simple brute force. It is also important that the size of the key be large enough to prevent brute-force attacks, in which the attacker simply tries every possible key until the correct one is found. Clearly, the size of the key that will minimally satisfy this requirement increases with time as the hardware available to an attacker becomes faster and the attack strategy becomes more sophisticated. A robust design must therefore use a key that is large enough to provide some margin of safety against these attacks. Alternatively, it must be a simple matter to increase the size of the key without causing existing applications to fail.

In the standard version of NTP, the originating and receiving systems use the same algorithm for constructing the authenticator. The only difference in the processing at the two end-points is that the originator simply attaches its result to the end of the message, while the receiver compares its result to the value that it was sent. The fact that the algorithms used by both systems are the same is an important point, because it means that the process used to compute the authenticator need not have an inverse. Apart from a number of theoretical advantages, there is one very important practical consequence – strong algorithms that have a unique inverse can be used in cryptography, and such algorithms often have export restrictions as a result. There are no such restrictions on one-way methods.

It is much more difficult to cope with an attacker who delays an otherwise legitimate message without altering it. The standard NTP algorithm estimates the one-way delay between the client and the server as one-half of the round-trip delay, which is measured using the data in the messages. An attacker who inserts an asymmetric delay into a network element therefore introduces a bias in the time of the client equal to one-half of this value. Rejecting packets

whose measured round-trip delays exceed some critical value can bound the magnitude of this problem. This strategy may not be optimum because it converts an attack that would have introduced a bias into the time of the client into one that effectively shuts down the client altogether. Rejecting packets in this way is therefore not a very effective defensive strategy, although it may be adequate in some situations.

If the clock in the client system need only be synchronized to the nearest second, for example, then it is difficult for an attacker to insert an asymmetric delay that is large enough to cause trouble while at the same time being small enough to escape detection by routine methods. It would be much more difficult to detect this sort of attack if the required accuracy were on the order of milliseconds, since, even under normal circumstances, fluctuations in the symmetry of the network delay are often of this order. Furthermore, these fluctuations often have a flicker-like spectrum at periods of a few hours, so that detecting a static bias can be very difficult in this situation. In the general case, therefore, it may be impossible to use the Internet to support authenticated time information with millisecond accuracy. The situation is more favorable with telephone-based connections or with systems connected to a local-area network (LAN) because the fluctuations in the circuit delay and in its asymmetry are much smaller to begin with. The potential accuracy of the synchronization process is therefore greater, and attacks are both harder to implement (because the physical circuits of the LAN and the telephone system are centrally managed) and easier to detect.

3. NTP AND SPOOFING

The symmetry of the authentication process used by NTP has a number of advantages, but it has one serious drawback. The client and the server use the same key, and there is therefore nothing in the authentication process to prevent a client from pretending to be a server. We can minimize the impact of such spoofing by giving each client a different key. While this minimizes the damage that any client can do, it substantially complicates the management of the keys themselves.

If any client can communicate with any server, then all of the servers must have tables of all of the possible client keys. This is possible in principle, although a successful attack on any server would then compromise the entire system by exposing all of the keys. A more robust arrangement would be to limit each machine to communication with only one or two servers. This arrangement is somewhat more susceptible to denial of service attacks, but more limited distribution of the keys results in an increase in security that more than offsets this problem.

Another possibility would be to use different keys for encryption and decryption. A number of "Public Key" algorithms of this type are in the literature. [2] Using such an algorithm would simplify the problem of managing and distributing the keys, but it might raise problems of possible export restrictions.

The increased complexity of the algorithms that are needed to realize public-key encryption and decryption make them impractical to use for authenticating the time messages themselves. However, they can be used to define a *session key*, which is then used to authenticate subsequent messages using a standard symmetric algorithm. The session key does not require the same level of protection as a static symmetric key, since it is only used for the duration of the current communications session.

4. CERTIFICATES OF AUTHENTICATION

Although all of the primary stratum-1 servers will be under the direct operational control of a national timing authority, the secondary stratum-2 servers will probably be operated by independent agencies. Protecting the messages transmitted between these systems is therefore not the whole story – in order to preserve traceability to the national time scale, we must also implement a method that can certify that the stratum-2 servers are operating properly. There is a similar requirement for a method to transfer a certificate between the stratum-2 servers and the next level of systems, which may actually apply time-stamps to transactions and which might also act as servers for another layer of clients.

The actual certificate can be transmitted using another network transaction, and it clearly must be protected using an authenticator that is similar to the machinery that we have already developed. The NTP protocol provides a class of messages that can be readily adapted to this function. Fluctuations in the time that it takes this message to travel from the server to the client are not critical, and the symmetry of the delay is not an issue. It is a simple matter to protect against re-play attacks by adding a serial number and expiration date to the certificate.

The period of validity of this certificate can be computed statistically. It is proportional to the ratio of the required time accuracy of the client to the Allan deviation of its clock oscillator, where the Allan deviation is evaluated at the averaging time corresponding the period of validity. The expiration date would probably include a safety factor to protect against unforeseen glitches and would therefore be somewhat sooner than the value derived from the statistics.

As an example, the Allan deviation of the clock oscillator used in a typical computer is about 10^{-6} at a period of 1d. This level of performance would support an accuracy of 1 s using less than one calibration cycle per day. The certificates used in this situation could have a shorter period of validity if desired. This period would be determined as a balance between the desire to detect failures as promptly as possible and the increased cost resulting from more frequent calibration cycles.

The symmetry of the authentication process would allow a client to forge its own certificate. The easiest way to deal with this is to have each server post its copy of the certificate so that it can be publicly viewed; another possibility would be to use public-key cryptography to sign the certificates. Using public-key cryptography seems like an unnecessary complication, but the machinery for this process is well known, and it can be added if needed. Even if none of these additional precautions is used, the forgery will be detected when the serial numbers of certificates issued on subsequent cycles disagree between the values expected by the server and the client.

5. THE COMPLETE PROTOCOL

We have now developed all of the pieces that are needed for specifying the complete authentication protocol. The client would initiate each transaction by sending a request for time to the server. (No matter who initiates the connection between the client and the server, it is important that the client initiate the conversation, because this choice requires that the client transmit its notion of the current time without any prompting from the server. A client whose notion of the current time is either in the future or too far in the past with respect to the clock on the server can be detected as unsynchronized without further analysis.) The Internet could be used to transmit this request if the required accuracy is only on the order of 1s. A dial-up telephone connection would probably be more appropriate for authenticated synchronization where significantly higher accuracy was required. (Even though telephone connections have delays that are both more stable and more nearly symmetrical than connections that use the Internet, the residual asymmetries in the delays in the circuit and the end-point hardware make it impractical to achieve an accuracy of significantly better than 1 ms. Realizing an accuracy of 1 ms requires a careful choice of modems, since many models have asymmetries in the delays between the inbound and outbound channels that can be as large as 10 ms.)

The server would respond to this message by adding its time to the packet and returning the message to the client. The interaction would be authenticated in both directions using the method described in RFC-2104 (or something equivalent)[3]. (The key used for authenticating this and all subsequent messages would be either the static key associated with this client-server pair or a session key that was negotiated at the start of the conversation using public-key methods.) Although this authentication procedure is not compatible with the existing NTP code, changing authenticating algorithms is a minor change to NTP, and no change to its basic message format would be required. If static keys were used, they would be transmitted between the server and the client using a secure channel. The design of this ancillary channel is not part of this authentication protocol. This additional channel would not be needed if session keys, negotiated using public-key methods, were employed.

In addition to the minor changes required to the existing NTP code, the most important consequence of this design is that casual associations between clients and servers would no longer be possible as at present. A server might still respond to a non-authenticated request for time, but the full machinery of the protocol would be available only to users who were previously known to the server and who had received a unique key to authenticate the requests. If session keys were used to authenticated requests, the public keys of the client and server would have to be transmitted using some ancillary secure method so that the negotiation of a session key could be performed.

At short periods (less than 1-2 s), the fluctuations in the system latency and in the symmetry of the delay contribute appreciable white phase noise to the measured time differences, and it is therefore advantageous to repeat this exchange several times and average the group of measured time differences. Our experience [4] with network-based synchronization algorithms suggests that using about 3 or 4 closely spaced packets results in a significant attenuation of the white phase noise without unduly increasing the cost of the synchronization. A simple way to realize this idea is to use 2 sets of 3 packets each, with the client and the server each acting as the originator for one set. The 2 systems can then exchange their estimates of the average time difference and network delay. Any disagreement between the two sets of estimates can provide an indication of a large asymmetry in the delay between the two systems. If the estimated time difference is within the specified limits, the server would then close the conversation by sending a certificate of authentication to the client. This certificate would contain the estimated average time difference, the uncertainty in this estimate, and the duration of its validity. The client could use the time

difference to discipline its clock or it could realize this discipline using some other reference source of time.

Although the individual messages that make up the complete protocol do not differ significantly from the formats used by NTP, the overall protocol is very different, and the standard NTP software will have to be modified to realize it.

6. BINDING THE TIME-STAMP

There is one final step in the process. The time-stamp and the transaction which references it must be bound together so that the association can be verified after the fact. This binding is currently often realized using a printed receipt, and the receipt might also include the stamp of a Notary Public in some cases. A number of digital versions of this idea are in the literature [5]. While this is an important step in the chain of traceability, the details of this step are outside of the protocol that we have discussed and are best left to the end-users of the time service.

7. CONCLUSIONS

The Internet is being used to support financial and commercial transactions, and these transactions often contain time-critical elements. The value of a single one of these transactions may be large enough to attract network-based attacks. I have suggested a method for protecting the messages that are used to synchronize the computers that are used to process these transactions. The method defines a protocol that is an extension of the currently defined Network Time Protocol. The changes to NTP that are necessary to support strong authentication are not large, but they probably require special-purpose software on both the client and the server to implement them. We do not see this as a serious obstacle to the implementation of authenticated time services, since it will probably be needed by only a very small fraction of the current users of network-based time services.

The accuracy of the authentication process will be limited by the asymmetry in the path delay between the server and the client. Attacks that attempt to bias the time of the client by intentionally increasing this asymmetry are easy to do and difficult to detect when the Internet is used as the transmission medium. This affects the accuracy of the authentication that can be realized using this transmission medium; the size of the offset is limited by the maximum asymmetry in the path delay that an attacker might be able to introduce without being detected. It is hard to provide an exact estimate of this value, but it probably could not be much greater than about 1 – 2 s. The time of the client would be biased by one-half of this value.

Dial-up telephone circuits have delays that are both more stable and more nearly symmetrical than logical circuits implemented using the Internet. In addition, it is more difficult to add a delay to one portion of the path without being detected, since delays longer than about 50 ms are very noticeable when the same circuit is used for a voice connection. These advantages should support substantially higher accuracy when the time messages are transmitted from the server to the client over dial-up telephone circuits, and it should be possible to provide an authenticated time service with a stability of better than 1 ms and an accuracy of 1-3 ms using this medium.

8. ACKNOWLEDGEMENTS

This work is supported in part by the National Science Foundation through grant NCR-9416663 to the University of Colorado.

9. REFERENCES

[1] D. L. Mills, "Internet Time Synchronization: The Network Time Protocol," IEEE Trans. Comm., vol. 39, pp. 1482-1493, 1991.

[2] Bruce Schneier, "Applied Cryptography, 2nd Edition, New York: John Wiley and Sons, 1996, pp. 461-482.

[3] H. Krawczyk, M. Bellare and R. Canetti, "HMAC: Keyed-Hashing for Message Authentication," RFC-2104, Network Working Group, February 1997. This document is available electronically using ftp and connecting to nis.nsf.net.

[4] Judah Levine, "Time Synchronization over the Internet using 'AUTOLOCK'," Proc. IEEE International Frequency Control Symposium, IEEE Catalog No. 98CH36165, pp. 241-249, 1998.

[5] Judah Levine, "Authentication, Time-Stamping and Digital Signatures," Proc. 27th Precise Time and Time Interval Planning and Applications Meeting, December, 1995, NASA Document CP-3334, pp. 439-445. See also B. Cipra, "Electronic time-stamping: the notary public goes digital," Science, vol. 261, pp. 162-163, 1993.