AUTHENTICATION, TIME-STAMPING AND DIGITAL SIGNATURES

Judah Levine
Time and Frequency Division
National Institute of Standards and Technology
Boulder, Colorado 80303

Abstract

Time and frequency data are often transmitted over public packet-switched networks, and the use of this mode of distribution is likely to increase in the near future as high-speed logical circuits transmitted via networks replace point-to-point physical circuits. Although these networks have many technical advantages, they are susceptible to eavesdropping, spoofing, and the alteration of messages enroute using techniques that are relatively simply to implement and quite difficult to detect.

I will discuss a number of solutions to these problems, including the authentication mechanism used in the Network Time Protocol (NTP) and the more general technique of signing time-stamps using public-key cryptography. This public-key method can also be used to implement the digital analog of a Notary Public, and I will discuss how such a system could be realized on a public network such as the Internet.

INTRODUCTION

Time and frequency transmissions – both the signals themselves and the interchanges of data that form the basis for national and international coordination – are generally transmitted with only moderate security and only cursory authentication. Although this openness has served us well up until now, the increasing importance of time and frequency information in many areas ranging from synchronous communications to the coordination of access to distributed databases means that the disruption of a time service may be very serious and costly. In addition, the continuing problems posed by computer viruses and worm programs should teach us that attacks on networks and computer systems are not necessarily motivated solely (or even primarily) by financial gain, and that attacking an important and visible system might be considered a challenge, and "just because it's there" could be enough of a motivation. It is important to begin now to consider how the security of our transmissions can be improved before any of our systems comes under a determined attack.

Jamming transmissions or cutting cables may have serious consequences by denying service to a particular group of users, but the effects tend to be localized and relatively easy to detect – even if correcting the problem can be both arduous and expensive. Data modifications that do not produce obvious disruptions or degradations are potentially much more serious. These changes are not readily detected; in the worst case the corrupted transmissions may be accepted as genuine for some time.

The various methods for protecting and authenticating data that I will discuss fall into two broad categories – methods that are intended to insure the integrity of existing time signals or data transmissions and methods that are potentially useful as ends in themselves in protecting or authenticating other kinds of time-sensitive messages and transactions.

SINGLE-KEY ENCRYPTION

The simplest way of protecting data is to add some form of check-sum to it and then encrypt either the complete message or just the checksum. Only someone who knows the key can alter the data without invalidating the original checksum. There are any number of methods that can be used for this purpose, including those that effectively "chain" messages together to prevent adding spurious messages or replaying older valid ones[1].

There are a number of technical difficulties with this system, including devising robust methods for dealing with noisy channels and lost messages, but the most serious practical difficulty is probably the distribution of the encryption and decryption keys. Whether these keys are implemented in software (using passwords or phrases) or in hardware (using an artifact such as a magnetic card), distributing keys requires a substantial investment of people and money and is probably never as secure as the advocates of such systems would claim.

In spite of these practical difficulties, the Internet-based Network Time Protocol (NTP) supports optional authentication and validation using single-key encryption[2]. When this mode is enabled, the transmitter of an NTP message computes an authenticator derived from the message itself and a secret key using the Data Encryption Standard (DES) in a mode similar to the "block-chaining" method specified by NIST[3]. The resulting checksum is then appended to the end of the message together with an integer specifying which of several possible keys has been used to compute it. The relationship between this integer and the actual key value must be sent to each receiver via an external, unspecified, secure channel. The receiver validates the message by repeating the same procedure and comparing the checksum it has computed with the value received in the message[4]. Although the DES standard defines both an encryption and a decryption algorithm, the decryption algorithm is not used by NTP so that the same software can be used at both ends. Furthermore, while both transmitter and receiver must use the same algorithm to authenticate the time packets, there is no reason why it must be the Data Encryption Standard, and the later versions of NTP support authentication using either DES or a variant of an algorithm called MD5[5].

In addition to the usual problems associated with distributing and maintaining the key database, authentication may degrade the accuracy of time-stamps transmitted using the NTP protocol. The checksum must be computed after the entire message has been constructed, and it therefore inevitably delays the transmission of the packet by an amount that depends on the speed of the processor and on its load. This delay appears as an increase in the out-bound network transit-time; depending on the mode of the association, there may be no corresponding inbound delay to preserve the symmetry. It is possible to make an approximate correction for the time needed to compute the checksum by inserting an estimate for this time into the NTP configuration file, so that the problem introduced by the authentication is due not so much to the delay it introduces, but to the unmodeled fluctuations in this delay caused by changes in load and similar factors.

Although this authentication method can be used by a private network of machines exchanging time messages only among themselves, it cannot be used to authenticate packets from a machine that is publicly available, such as the NIST primary time server. (The authentication in this case is only for the benefit of the client machines – the server itself has no need for authentication since it is synchronized to UTC via external means and does not accept network-based synchronization information.) The key used by the receiver to verify the authenticity of a packet is the same as the key used by the transmitter to produce it, so that the key used by a primary server must be widely known if the authenticity of its time packets is to be verified by its clients. But once the key is widely known, numerous other machine can use it to generate "authentic" packets as well, so that its use provides little or no security. In practice, therefore, the authentication scheme incorporated into NTP is only useful in validating transmissions among a group of peers which are under the control of a single management entity.

The NTP protocol also supports an authentication mechanism based on the Internet address of the time source – a machine can be configured to accept information only if the network address of the source matches one of the entries in an internal table[4]. Although the time needed for this check can be made outside of the primary packet-exchange loop so that it does not degrade the transit-time estimate, the procedure can be quite lengthy if many addresses must be examined. As a consequence, this procedure is useful to safeguard a relatively small group of machines. It does not provide fool-proof protection even in this case because of the increasingly common practice of "ip-address spoofing."

DUAL-KEY ENCRYPTION

Dual-key encryption (often called public-key encryption) is designed to overcome some of the problems with conventional single-key methods that I discussed in the previous section. A dual-key system uses two different keys (and usually two complementary procedures as well) to perform encryption and decryption. These ideas can be realized in a number of different ways, but all of them share the same basic properties[6]. A message, M, is encrypted using an encryption key e and an encryption procedure E to produce a cipher text C. Thus

$$C = E(e, M). \tag{1}$$

There exists a decryption procedure D and a key d so that applying them to the cipher text recovers the original message:

$$M' = D(d, C), \tag{2}$$

where M′ is identical to M if and only if the key d is the cryptographic inverse of the key e. As with most cryptographic systems, the procedures E and D are well known; the security lies in the choice of the keys e and d. Although the two keys must be related mathematically, it is computationally infeasible to compute one from the other in any finite time. Different algorithms realize this secret linkage differently: in some cases the keys are the prime factors of a very large number, and the security rests on the difficulty of computing such factors. In other systems, the linkage is realized through discrete logarithms modulo a large integer, and the security comes from the difficulty of inverting such procedures.

The dual-key system may be used in two different ways. If the encryption key is made public and the decryption key kept secret, then anyone can encrypt a message, but only the holder of the secret key can read the resulting cipher text. This configuration could be used to encrypt electronic mail messages or data transmissions, for example, so that only the intended recipient could read them[7]. If, on the other hand, the encryption key is kept secret and the decryption key is made public, then only the holder of the key can encrypt a message but anyone can read it. The existence of the encrypted text authenticates the authorship of the corresponding plain-text message, because only the holder of the secret key could have computed the encryption. There is no need to archive the message or the cipher text – the authentication can be verified by anybody, since both the decryption algorithm and the key are publicly known. Displaying the two versions is then a form of "digital signature" – the holder of the secret key can use the display as proof of authorship, and a third party can use the display to prevent the message from being repudiated. (Preventing a valid message from being repudiated by its author often has important legal and commercial applications[6].) Although I would focus on the authentication aspects of a digital signature in the current discussion, both the encryption and authentication functions would be useful in the time and frequency business.

The dual-key system changes the problems associated with distributing the keys, but it does not eliminate them. The secret key must remain secret, of course, but the more difficult issue is insuring that the public key, which must be widely available by definition, is not altered surreptitiously. A random alteration is the digital equivalent of jamming a radio signal – it may break the system locally, but is relatively easy to detect. Altering the public key so that it is the inverse of another private key is a more serious problem. If a third party with malicious intentions can manage to replace the legitimate public key with one that is the complement of his own private key, then a variety of more sophisticated attacks becomes possible, at least some of which have been discussed in the literature[8]. One way to address this problem is to have a central trusted repository for the public keys, but this simply pushes many of the problems we have been discussing into the design of this repository – it must be very secure and robust because it will become both an attractive target to attack and a single point of failure. The methods it uses to authenticate its responses to legitimate requests for public keys must be carefully studied as well to minimize the probability of undetected spoofing.

Dual-key systems operating in the "digital signature" mode can provide a mechanism for authenticating time signals in principle, but there are a number of practical difficulties that have prevented their widespread use. The key distribution problems discussed above are a significant difficulty. In addition, the time needed to compute the digital signature may significantly affect the accuracy of the transmissions. The computations are much more intensive than those required for single-key methods, and computing a signature may take an appreciable fraction of a second – even on a fast processor with optimized code. As with single-key methods, the average delay introduced by the computation could be estimated and a correction for it could be incorporated into the software, but the load fluctuations are likely to be more important because the correction itself is significantly larger. Furthermore, the complexity of the calculations can place a heavy load on a primary time server, which may have to respond to more than 10 requests/second during peak load.

A DIGITAL TIME-STAMP SERVICE

There are many situations where it is important to be able to prove that a document in digital format existed on a certain date and time in its current form. Examples include the disclosure of inventions, and many business transactions where time is a factor. The digital-signature algorithms can be combined with time signals in digital format to provide a publicly available time-stamp service for such documents that has many of the features of a Notary Public. In addition, the digital time-stamp service has a number of features that are not available with a traditional Notary including authenticating "documents" that are not text at all (such as a digitized photograph, a musical composition, or a compiled computer program) and providing authentication for a document without the need for revealing its contents to the authenticating authority.

The general principles of this system are derived from the dual-key cryptography discussed above: The document to be signed is submitted to a timing laboratory such as NIST; the NIST server adds a time-stamp and then signs the result using its private key; the resulting compound document can be verified by anyone who has the corresponding public key without involving NIST at all – only the public key and the appropriate algorithm are required. The mathematics underlying the signature algorithm guarantees that the document, the time-stamp, and the signature originated from NIST and that neither the document nor the time-stamp could have been altered without invalidating the signature. Since the signature is an explicit function of the compound document, neither it nor the time-stamp can be reused on another document. Finally, the authentication of a signature can be performed independently of the original signatory and there is no need for a central repository of signed documents.

The basic design would not necessarily detect a "replay" attack in which a valid document is re-transmitted a second time. If such attacks are a concern (as they might be with the message, "Deposit $100 to my account ...") then simply adding sequence numbers to each message will detect the second message as a duplicate.

This general procedure can be strengthened and generalized in a number of important ways:

1. Using a technique known as "hashing" it is possible to eliminate the need to submit the document itself to the signature authority. A "hash" function is a one-way function that accepts a digital string of characters and computes a fixed-length output value based on the entire string. The check-sum characters often used in digital transmissions and the last character of the numbers used to identify products to a supermarket scanner are

simple examples. The hash computation is a unique function of its input in the sense that changing any character of the input changes the hash, but it possesses no inverse — the input cannot be constructed knowing the hash output except by trial and error. Since the hash function is a unique function of its input, signing it is equivalent to signing the original, with the advantage that the original is not limited to printable text but can be a digitized photograph, a digital recording or any string of digital numbers. The exact contents need not be revealed to the signature authority. As a result, the hash values can be transmitted over insecure public channels so that the digital signature system can be implemented using ordinary electronic mail.

2. If the system that receives the messages and adds the time-stamp is also a time-server for the network, then the accuracy of its time can be verified at any time (using NTP, for example) independently of the signature algorithm. The NIST time-servers, for example, receive 50000+ requests for time every day, so that any attempt to alter the time used for the stamping process would be widely detected almost immediately.

3. The security of the system ultimately depends on the security of the private key used to sign the documents. If this private key is stored on the machine that is visible on the public network, then a successful network-based attack on this machine might reveal the secret key and compromise the entire system. This problem can be addressed by storing the key on a second "back" machine that is connected to the time-stamp machine via a private link that is not visible from the network. The two functions of signing and time-stamping are divided in this configuration, with the front machine adding the time-stamp and the back machine computing the signature. While the time-stamp machine must operate in "real time" signing documents as soon as they arrive, the back machine that computes the signature does not have this requirement. It can operate in a batch mode at discrete intervals, taking messages from an input queue on the front machine and returning them to an output queue there. The signed messages on the output queue can be returned to the sender or can be forwarded to a third party as required.

4. The concept of using two machines can be generalized to improve the reliability of the system by constructing several "front" and several "back" machines. The various "front" machines might accept messages in a number of different formats including electronic mail, scanned images, and binary digital files of unspecified format.

As in the previous discussion, the overall structure can be realized using a number of different algorithms, both in the computation of the hash function and in the actual signature process itself. The modular nature of the procedure makes is relatively simple to change to a new algorithm without breaking the overall system.

CONCLUSIONS

The increasing use of public packet-switched channels to transmit time and frequency signals and data raises concerns that these transmissions may be intercepted or modified by third parties with malicious intent. We have discussed a number of ways of safeguarding and authenticating messages sent over public networks; none of them is widely used at this time and all have some practical drawbacks. In addition to the issues we have already discussed, some of the algorithms are proprietary and can only be used with a rather expensive license and others are considered sensitive and cannot be exported outside of the US without a special export license[9].

The experiences of the PC and Internet communities is that financial gain is not necessarily the only motive for attacking network-based services. Given these experiences, it is probably only a matter of time before a serious assault is attempted. It is probably useful to begin to think about these issues now while we can do so without the threat of an imminent network-based attack hanging over our deliberations.

As more and more documents are transmitted and stored in digital format, there will be an increasing need for the digital equivalent of a Notary Public. The public-key algorithms that we have discussed can be combined with precise digital time signals to realize such a system at only modest cost.

REFERENCES

[1] B. Cipra 1993, *"Electronic time-stamping: the notary public goes digital,"* **Science, 261**, 162-163.

[2] D.L. Mills 1991, *"Internet time synchronization: The Network Time Protocol,"* **IEEE Trans. Comm., 39**, 1482-1493.

[3] Federal Information Processing Standard 46: The Data Encryption Standard, 1977, National Bureau of Standards.

[4] D.L. Mills 1992, *"Network Time Protocol (Version 3), specification, implementation and analysis,"* Network Working Group Report RFC–1305.

[5] R. Rivest 1992, *"The MD5 message digest algorithm,"* Network Working Group Report RFC-1321.

[6] J. Nechvatal 1991, *"Public-Key Cryptography,"* NIST Special Publication 800-2.

[7] S. Garfinkel 1995, *"PGP – Pretty Good Privacy,"* Sebastopol, California, O'Reilly & Associates, chap. 3.

[8] B. Schneier 1994, **Applied Cryptography** (John Wiley & Sons, New York), chap. 3.

[9] Reference [8], chap. 18, pp. 448-454.