M.I.T. Laboratory for Computer Science
Computer Systems Groups

May 27, 1981
Request for Comments No. 208

HAZARDS OF FILE ENCRYPTION

by J. H. Saltzer

Abstract

Encryption of stored files is an attractive way of separating responsibility for storage management from that of privacy. However, blind application of encryption techniques designed for use in communication systems, where messages are evanescent, decryption occurs immediately, and random access to data is never required, can lead to a storage system design that has less privacy, reliability, and function than intended. This paper discusses some of the special problems of file encryption that arise because the storage environment and the communications environment are different, and offers some suggestions for their solution.

Introduction

The application of cryptographic techniques to achieve privacy and authenticity of communication has a long history, including experiences both good and bad[1,2]. This long collection of experiences has produced a gradual sophistication in the understanding of the design considerations involved. Recently, with the addition of the techniques of public key encryption[3,4] and key distribution protocols[5], it appears that the basic ideas of cryptography can be extended to multipoint data communication networks. Less experience with these ideas has been obtained, so the

WORKING PAPER -- Please do not reproduce without the author's permission and do not cite in other publications. This paper has been submitted to the 8th ACM Symposium on Operating Systems Principles.

relevant design considerations are not yet so clear. Nevertheless, the multipoint communications application is similar enough to the point-to-point experience that one expects most of that experience to carry over. On the other hand, new applications of cryptography are being explored, including encryption of stored files[6,7,8]. Unfortunately, blind transfer of a successful technique to a domain for which it was not originally designed can yield surprises. There are several hazards in file encryption that are not present in the same form or with the same priority in communication systems; these hazards must be recognized by proposers or implementors of file encryption systems. This paper points out several of those hazards, and offers some suggestions for avoiding or coping with them.

Before looking at the hazards, let us review what benefit one might hope to achieve by applying encryption to stored files. The chief benefit seems to arise from an extremely powerful system modularizing effect: if one encrypts the data, its privacy is secured without question of how the physical storage is later managed. All aspects of storage management are then relieved of the need to consider privacy in their actions. Management of detachable storage media may be especially simplified. Thus, for example, no-longer-needed backup tapes containing file copies can be recycled without special steps to erase them. A misbehaving disk drive can be turned over to a customer engineer for repair without worrying about whether its contents are sensitive. A disk pack can be sent to an outside service for cleaning. Inside the operating system, the size of the trusted "kernel," that is the collection of supervisor procedures that are in a position to compromise privacy, can be smaller, since the storage management programs do not need to be included. In the extreme, if encrypting and decrypting of files are done outside, rather than inside the computer system, one can make a reasonable argument that privacy can be had in a multi-user computer system that provides no privacy-achieving mechanisms of its own. Of course this last approach also prevents the shared computer system from doing any processing of the data. However, there are sensible applications for such non-processing data storage system, such as shared storage repositories in a network of personal desktop computers. Data may be enciphered by the desktop computer in a private environment, before being sent to the shared storage repository where it is expected that storage costs are lower, and long-term reliability is higher. Exactly such strategies for privacy have been recently proposed for distributed storage systems[8].

With this appreciation of the possible benefit, what then are the new and unusual hazards of file encryption that might require a different set of design considerations as compared with an encrypted communication system? They can be broadly categorized according to three ways that a stored file system differs from a communication system:

1) The storage of a file system is intended to be persistent, while the presence of information in a communication system is evanescent.

2) At the time decryption occurs in a file system, the original, unencrypted source text may not be available anywhere for comparison. In a communication system the originator usually holds the original message text until after the recipient has acknowledged that decryption was successful.

3) In a file one often wants random access to interior parts of the file (e.g., records). In a communication system, data departs and arrives in a stream, allowing sequential encryption and decryption.

These three differences between file storage and communication can strongly affect the details of design of the cryptographic system. Each of the three differences is the source of several hazards, as discussed in the following three sections.

Persistence

File storage is normally used for data that will be left in place for some length of time. This persistence produces three effects that are somewhat different from the corresponding situation in a communication system where data is encrypted, transmitted, and immediately decrypted.

The first effect we might name the "sitting duck" effect, thinking of the relative ease of shooting a duck sitting in a pond compared with one flying overhead. To attack a specific conversation in a communication system, one must first intercept the conversation, which requires being in the right place at the right time. The individual messages are evanescent, so an attacker must capture a copy as it flies by, accurately, perhaps without being certain of what he has captured. In contrast, an encrypted file may sit for days, weeks, or months in one place, allowing the attacker to choose his time and weapons, and try again if the first attempt produced a copy of the wrong file or had errors in it. Although evanescence of messages is not usually explicitly acknowledged as increasing the work factor in communication systems (it has been mentioned in connection with packet switching networks with dynamic routing[9]) it nevertheless is part of the accumulated experience with successful communication systems and protocols, and it is difficult to judge how much extra strength is needed elsewhere (for example in the cryptographic transformation itself) to compensate for persistence in the file storage system. Conservatively, though, one would expect that a stronger cryptographic transformation ought to be specified for file storage.

A second effect, similarly difficult to evaluate, relates to the time value of the information being protected, A standard criterion for cryptographic systems used for communications is the length of time that the transmitted information needs to be protected as compared with the minimum time a well-equipped attacker needs to cryptanalyze it. For example, a message that needs to be private for only a day or so could be adequately protected by a cryptographic system that can be systematically penetrated with a week of effort. If one looks at the distribution of time values of a set of messages on the one hand and a set of files on the other hand, one might expect to find some important differences. Messages probably have predominently short time values, measured in days or weeks, while files may need to be kept private for relatively long times, perhaps months or years. These time values of messages should be assumed to have been factored in to the experience of communication cryptographic system designers. Thus, again, it may be necessary to insist that cryptographic transformations for use in file storage have higher work factors than transformations intended for use in communications.

Neither of these effects, the "sitting duck" and the time value, are absolute, but rather are subtle differences of emphasis between file storage and communication, and their biggest impact must come in understanding the meaning of a "certification" of a cryptographic system as adequate for a particular application. Since the average application user is unlikely to have the diligence or mathematical expertise to analyze a cryptographic system himself, he must rely on the experience of others, in the form of a certification. The point here is that that experience is extensive in the area of communications, where evanescence and short time values are part of the environment, and almost non-existent in storage, with persistence and long time values. So certification based on communication experience may be less significant than it appears on the surface.

The third and last effect of persistent storage is that one normally assumes in the design of a cryptographic system that cryptographic keys will eventually be compromised, through human blunders and miscalculation. In a communications system, one plans to change keys whenever a compromise is suspected, as well as periodically with a frequency determined by experience. Put another way, key management, which involves human operations, must be sufficiently disciplined to match the time value of the information. For the case of long term file storage, it is difficult to imagine how to assure long term lack of compromise of keys. Unfortunately, the alternatives are not very enticing either. If a key is accidentally compromised, (or compromise is suspected) re-encipherment of affected files seems to be required, presumably a costly operation. In addition, re-encipherment of a file gives a cryptanalyst two enciphered versions of the same plain text, which may lower the work factor. This area has apparently not yet been carefully explored, and some opportunities may exist for ingenuity.

Time of decryption

In a file storage system, the time scenarios are very different from those of a communications system, and these scenario differences lead to profound differences in the effects of various kinds of failures. A communication system is usually designed under the assumption that decryption happens shortly after encryption. That is, the recipient normally decrypts the message, establishes that the transmission went well, and acknowledges it, all while the originator maintains his copy of the message. Only after acknowledgement of successful transmission does the originator consider destroying his copy of the message. (Even then the unencrypted plain text may be logged.) If trouble is detected, the original text is available for retransmission, and recovery from errors is designed with this option in mind.

In the file storage system, a whole different scenario is usually envisioned. The data is encrypted and stored, and then the original is destroyed. During the term of storage, the unenciphered data does not exist. Later, perhaps much later, decryption must be accomplished without the possibility of going back to the original for help. If errors never happen, this difference in scenario would be of little concern. Unfortunately, errors do happen. Some possible errors include:

- the enciphering engine may have a hardware error,
- the storage system may drop some bits during storage,
- the key may not have been correctly recorded,
- the human user may have made some mistake, such as encrypting the file twice or using the wrong key.

If the corresponding error occurs in a communication system, recovery can be quick and easy, because the original data is available for retransmission as soon as the problem is discovered and repaired. In the file system no such appeal to the original data is available. In recovery terms, "backward" error recovery, in which one goes back to an earlier state and tries again, is not feasible, so one must invent a whole new collection of "forward" error recovery techniques, in which the original encryption of data and key management are done so carefully that there is little chance of a problem occurring at decryption time.

Such forward error recovery techniques can certainly be invented. For example, one might encrypt using two parallel encryption devices, comparing their output. After writing the data on the disk one might read it back and decrypt it to verify its correctness, before destroying the original. In doing so, one should be certain to obtain the decryption key from the exact physical place it will be obtained in the future, rather than from some cache that might not properly be written into backup storage. Keys might be stored redundantly (with some concern as to whether or not this redundant storage represents excessive exposure.) The human factors of the system must in any case be designed to tolerate all imagineable (and unimagineable) blunders and missteps on the part of the user. All these things are possible, perhaps even straightforward. But they represent differences with the communication system design, and experience must be gained to learn how to provide forward error recovery that is effective at the same time that it is inexpensive and unobtrusive.

Random access

The third major way in which a communication system and a file storage system differ is that data in a communication system is almost always received in a sequential stream, so the encryption system is designed assuming that property. In contrast, in a file system there is often a requirement for rapid access to randomly chosen records in the middle of the file. This random access requirement leads to a need to redesign the basic encryption protocol. (This consideration therefore takes us temporarily into some of the more technical aspects of encryption.)

The reason for incompatibility between sequential encryption and random access becomes apparent in considering what happens if one applies sequential encryption to a file. In sequential encryption of a stream of data, the transformation applied to the next part of the stream depends not only on the key in use, but also on the preceding data in the stream. (This interdependence of one part of the stream with the next increases privacy substantially in the common case that data values repeat; if done carefully, interdependence can destroy any opportunity for attack by frequency analysis.) But the corresponding decryption must also be done sequentially, starting with the first bit of the stream. If one encrypted a file as though it were a stream of data, access to a record in the middle would require reading and decrypting the file from the beginning; a change to the file would require re-encrypting and rewriting the file from the point of change all the way to the end.

One could instead use a block-encryption scheme for files, for example applying the U.S. Data Encryption Standard[10] (which transforms a 64-bit block into another 64-bit block) to each block of the file independently. With this approach, one could read any random block of the file and decrypt it directly, and change, reencrypt, and rewrite random blocks. But this approach makes frequency analysis a possibly productive activity for an attacker, and comparison of a block-encrypted file before and after update almost perfectly exhibits the pattern of changes made by the update. An intermediate approach might be to divide the file into regions that are each to be enciphered sequentially but independently of one another. The region would be large enough not to be repetitive but small enough that it is not too painful to read an entire region to obtain access to an item stored therein. The trouble with this approach is that the choice of the best region size tradeoff between privacy and accessibility probably depends on the application, yet the application designer may not be technically equipped to judge the privacy impact.

More sophisticated techniques might be proposed to salvage the random access capability that comes with enciphering small blocks. For example, one might perform an "exclusive or" of the data in a block with its own address before enciphering it. Later, after decipherment, the "exclusive or" can be easily undone to recover the data, and enciphered repeated data stored in different locations will not look repetitive. But one should analyze that kind of technique in light of the particular encipherment algorithm to be used (and with some suspicion) because some other common data pattern may suddenly look repetitive, or the sequential nature of the addresses may still be apparent even after encryption, and in any case an attacker can still easily discover the pattern of changes made in updating a file. Random access is still a subject for research[11].

This set of considerations leads one into reconsideration of the basic encryption strategy itself, unfortunately opening up a world of esoteric mathematics and related cryptology. That reconsideration may turn out to be productive, but one must realize in doing it he has departed from the well-tested, high-confidence world of communication cryptography, and regaining of confidence will require new, long experience. So, once again we have identified an area in which the file storage application and the communication application differ enough that one cannot automatically assume that the communication cryptography techniques can be directly applied.

We should also note, however, that there are storage applications, such as the remote, low-cost storage of files in a distributed system with a high-speed communications network, where reading and rewriting the entire file may be quite an acceptable operation; when random access is not a requirement this third hazard seems to vanish from the scene.

Conclusion

We have identified several ways in which the mechanics of file encryption differ significantly from the mechanics of communication stream encryption. In particular:

- files tend to last longer than messages, so one must be sure the encryption system adequately takes into account their long-term vulnerability, their long time value, and the need to change compromised keys.
- files are deciphered at a time when the original copy no longer is available, so all failure recovery must be planned in advance.
- files sometimes require random access, which calls for encryption techniques that do not depend on sequential coding for security.

One might try to draw from these observations a conclusion that it is a mistake to apply encryption to files. Such a conclusion does not seem to be warranted. In light of the potential benefits of file encryption in separating privacy from storage management, especially in distributed systems, it seems appropriate to pursue the idea. However, this pursuit ought to be with the caution that the environment is different enough from that of the communications world that careful re-evaluation is needed at every step until a body of field experience has been acquired. One cannot assume that simple adaptation of the standard communication encryption techniques will provide the intended level of privacy, reliability, or function.

Acknowledgements

Although the design considerations described here apparently never have been collected in one place before, few of them are new. They seem most often to have been communicated as oral folklore rather than in published papers. I am indebted to many people who over a period of 15 years have each been willing to impart their share of the folklore to me. The names I recall are Dennis Branstad, Daniel Edwards, Edward Glaser, Martin Hellman, Steven Kent, Joel Oseas, Brian Randell, Ronald Rivest, Nathaniel Rochester, Michael Schroeder, Walter Tuchman, and Gerald Walter.

References

[1] Kahn, D., The Codebreakers, MacMillan, New York, 1967.

[2] Johnson, B., The Secret War, British Broadcasting Corporation, London, 1978, Chapter 6.

[3] Diffie, W., and Hellman, M.E., "New Directions in Cryptography," IEEE Trans. on Info. Theory IT-22, 6 (November, 1976) pp. 644-654.

[4] Rivest, R.L., Shamir, A., and Adleman, L., "A Method of Obtaining Digital Signatures and Public-Key Cryptosystems," Comm. ACM 21, 2 (February, 1978) pp. 120-126.

[5] Needham, R.M., and Schroeder, M.D., "Using Encryption for Authentication in Large Networks of Computers," Comm. ACM 21, 12 (December, 1978) pp. 993-999.

[6] Denning, D.E., and Denning, P.J., "Data Security," Computing Surveys 11, 3 (September, 1979) pp. 227-249.

[7] Ehrsam, W.F., et al., "A Cryptographic Key Management Scheme for Implementing the Data Encryption Standard," IBM Sys. Journal 17, 2, 1978.

[8] Reed, D.P., "Implementing Atomic Actions on Decentralized Data," presented at ACM 7th Symposium on Operating Systems Principles, December, 1979. To be published in the Comm. ACM.

[9] Baran, P., "Security, Secrecy, and Tamper-free Considerations," On Distributed Communications, 9, Rand Corp. Tech. Rep. RM-3765-PR, 1964.

[10] "Data Encryption Standard," Federal Information Processing Standards Publication 46, National Bureau of Standards, January 1977.

[11] Kent, S.T., "Protecting Externally Supplied Software in Small Computers," Ph.D. thesis, M.I.T. Dep't of Electrical Engineering and Computer Science, September, 1980. Also Laboratory for Computer Science Technical Report, TR-255, September, 1980.