Protocols for Public Key Cryptosystems

Conference Paper · April 1980

DOI: 10.1109/SP.1980.10006 · Source: DBLP

1 author:

Ralph C. Merkle
Institute for Molecular Manufacturing
74 PUBLICATIONS **6,614** CITATIONS

PROTOCOLS FOR PUBLIC KEY CRYPTOSYSTEMS

Ralph C. Merkle

ELXSi International

Sunnyvale, Ca.

Abstract

New cryptographic protocols which take full advantage of the unique properties of public key cryptosystems are now evolving. Several protocols for public key distribution and for digital signatures are briefly compared with each other and with the conventional alternative.

1. Introduction

The special strengths of public key systems are briefly considered by examining cryptographic protocols for key distribution and digital signatures using both public key and conventional systems.

The reader is assumed to be familiar with the general ideas behind public key cryptosystems, as described in [1,10].

For many of the following examples we assume there are two communicants, called A and B, and an opponent E. A and B will attempt to send secret messages and sign contracts, while E will attempt to discover the keys, learn the secrets, and forge contracts. Sometimes, A will attempt to evade a contract he signed with B, or B will attempt to forge A's signature to a new contract.

A and B will need to apply one way functions to various arguments of various sizes, so we assume we have a one way function F which can be applied to arguments of any size and produce a fixed size output. For a more complete discussion of one way functions, see [2,9,13,19].

This work was partially supported under NSF Grant ENG 10173, and much of the work was done at Stanford University ISL. The author would also like to acknowledge the support of BNR Inc, where much of the work reported here was done. An extended version has been submitted to CACM.

2. Centralized Key Distribution

Centralized key distribution using conventional encryption functions was the only reasonable method of handling key distribution in a multi-user network environment before the discovery of public key distribution methods. Only conventional encryption functions need be used, which presently offers a performance advantage. (Presently known public key systems are less efficient than conventional cryptographic systems. Whether or not this will continue is not now known. Discovery of new public key systems seems almost inevitable, and discovery of more efficient ones probable.)

In centralized key distribution, A, B, and all other system users somehow deposit a conventional cryptographic key with a central key distribution center. If A wishes to communicate with B, the key distribution center will send a common (session) key to A and B using the previously agreed on central keys. A and B can then communicate with no further assistance from the key distribution center.

This protocol is simple and requires only conventional encryption functions. Its use has been defended in the literature [17,18,20].

The major drawback of this protocol is its vulnerability to both centralized loss of security and centralized loss of function. Theft of the central keys, or bribery of personnel at the central site will compromise all users of the system. Similarly, destruction of the central keys destroys the key distribution mechanism for all users.

The security and reliability of centralized key distribution can be increased by using two or more centers, each with its own keys [1]. Destruction or compromise of a single center will not affect the other centers.

Security can also be improved if all the user keys are encrypted with a master key by the center. The master key must still be stored securely (and suitable provision made for its backup), but the (encrypted) user keys can be stored anywhere. This approach is used by IBM [23].

This protocol does not fully solve the key distribution problem: some sort of key distribution method must be used between each user and the center to establish the original keys. This problem is nontrivial because no electronic communications can be used and inexpensive physical methods, e.g., registered mail, offer only moderate security. The use of couriers is reasonably secure, although more expensive.

3. Simple Public Key Distribution

This is the most basic application of public key systems [1,5,6,7,8]. Its purpose is to allow A and B to agree on a common key k without any prior secret arrangements, even though E overhears all messages. A randomly computes enciphering and deciphering keys E_A and D_A, and sends E_A to B (and E). B picks the random key, k, and transmits $E_A(k)$ to A (and E). A computes $D_A(E_A(k)) = k$. A then discards both E_A and D_A, and B discards E_A. The key in future communications is k. It is used to encrypt all further messages using a conventional encryption function. Once A and B have finished talking, they both discard k. If they later resume the conversation the process is repeated to agree on a new key k'.

This protocol is very simple, and has a great deal to recommend it. First, no keys and no secret materials exist before A and B start communicating, and nothing is retained after they have finished. It is impossible for E to compromise any keys either before the conversation takes place, or after it is over, for the keys exist only during the conversation.

The disadvantage of this protocol is that E might actively interfere with the exchange of keys. Worse yet, E can force a known k on both A and B.

4. Authenticated Public Key Distribution

The now classic protocol [1] for secure and authenticated communications between A and B is: A and B generate E_A and E_B and make them public, while keeping D_A and D_B secret. The public enciphering keys of all users are entered in a public file, allowing easy and authenticated access to E_X for any user, X.

If A and B wish to agree on a common key k, then each sends a (session) key to the other by encrypting it with the others public key. The two keys thus agreed on are combined and used to encrypt further messages.

At the end of this protocol, A and B have agreed on a common key, k, which is both secret and authenticated.

This protocol suffers from two weaknesses. First, entries in the public file might be altered. This can be dealt with both by good physical security, or by using new protocols (see sections 5 and 6) for authenticating the entries in the public file.

Second, secret deciphering keys can be lost. This problem must ultimately be solved by good physical security.

5. Public Key Distribution with Certificates

Kohnfelder [3] first suggested that entries in the public file be authenticated by having a Central Authority (CA) sign them with D_{CA}. He called such signed entries certificates.

The protocol with certificates is the same as the authenticated protocol, except that A and B can now check the entries in the public file by checking each other's certificates. This protocol assures A and B that each has the other's public enciphering key, and not the public enciphering key of some imposter.

The security of this protocol rests on the assumptions that the secret deciphering keys of A, B, and CA have not been compromised; that A and B have correct copies of E_{CA} (to check the signed certificates); and that CA has not issued a bad certificate, either deliberately because it was untrustworthy, or accidentally because it was tricked.

E_{CA} can be published in newspapers and magazines, and sent over all available communication channels: blocking its correct reception would be very difficult.

If D_{CA} is compromised, then it is no longer possible to authenticate the users of the system and their public enciphering keys. The certificates are now worthless because the (unauthorized) person who has learned D_{CA} can produce false certificates at will.

6. Public Key Distribution with Tree Authentication

Key distribution with certificates was vulnerable to the criticism that D_{CA} can be compromised, resulting in system wide loss of authentication. This problem can be solved by using tree authentication [13].

Again, this protocol attempts to authenticate entries in the public file. However, instead of signing each entry in the public file, this protocol applies a one way hash function, H, to the entire public file. Even though H is applied to the entire public file, the output of H is only 100 or 200 bits long. The (small) output of H will be called the root, R, of the public file.

If all users of the system know R, then all users can authenticate the correctness of the (whole) public file by computing R = H(public file). Any attempt to introduce changes into the public file will imply R $\neq$ H(altered public file), an easily detected fact.

This method effectively eliminates the possibility of compromising D_{CA} because no secret deciphering key exists.

Because the public file will be subjected to the harsh glare of public scrutiny, and because making alterations in the public file is effectively impossible after it has been published, a high degree of assurance that it is correct can be attained.

This method is impractical as stated. Fortunately, it is possible to selectively authenticate individual entries in the public file without having to know the whole public file by using Merkle's "tree authentication," [13].

The essence of tree authentication is to authenticate the entire public file by "divide and conquer." If we define $\underline{Y}$ = public file = $Y_1, Y_2, \ldots Y_n$, (so the ith entry in the public file is denoted Y_i, and B's entry is Y_B); we can define H(public file) = H($\underline{Y}$) as:

H($\underline{Y}$) = F(H(first half of $\underline{Y}$),
H(second half of $\underline{Y}$))

Where F is a one way function.

If A wishes to confirm B's public enciphering key, then A need only know the first half of the public file, (which is where Y_B appears) and H(second half of public file) which is only 100 bits long. A can compute H(public file) knowing only this information, and yet A only knew half the entries in the public file.

In a similar fashion, A does not really need to know all of the first half of the public file, for

H(first half of public file) =
F(H(first quarter of public file),
H(second quarter of public file))

All A needs to know is the first quarter of the public file (which has Y_B), and H(second quarter of public file).

By applying this concept recursively, A can confirm Y_B in the public file knowing only R, $\log_2$ n intermediate values, and Y_B itself. The information needed to authenticate Y_B, given that R has already been authenticated, lies along the path from R to Y_B and will be called the authentication path.

These definitions are illustrated in figure 1, which shows the authentication path for Y_5.

For a more detailed discussion the reader is referred to [13].

Using tree authentication, user A has an authentication path which can be used to authenticate user A's public enciphering key, provided only that R has already been authenticated. An "authentication path" is a new form of certificate, with E_{CA} replaced by R.

This protocol can only be compromised if: D_A or D_B is compromised, or if R is not correctly known by A or B, or if there is a false and misleading entry in the public file.

The latter two are easily detectable. If either A or B has the wrong R, they will be unable to complete the protocol with any other legitimate user who has the correct R, a fact that will be quickly detected.

Because the public file is both open to public scrutiny and unalterable, false or misleading entries can be rapidly detected. In practice, a few users concerned with correctness can verify that the public file satisfies some simple global properties, i.e., each user name appears once and once only in the entire public file; individual users can then verify that their own entry is correct, and need not bother examining the rest of the public file.

The only practical method of compromising this protocol is to compromise D_A or D_B. A user's security is thus dependent on himself and no one else.

7. Digital Signatures

The use of public key cryptosystems to provide digital signatures was suggested by Diffie and Hellman [1]. Rivest, Shamir and Adleman [8] have suggested an attractive implementation. Signature techniques based on methods other than public key cryptosystems have been suggested by Lamport and Diffie [1,24], Rabin [15], and Merkle [13].

Digital signatures, whether based on conventional encryption functions, on public key cryptosystems, on probabilistic computations, or on other techniques share several important properties in common. These common properties are best illustrated by the following now classic example.

A wishes to place a purchase order with his stock broker B. A, on the Riviera, cannot send a written order to B in New York in time. All that A can quickly send to B is information, i.e., a sequence of bits, but B is concerned that A may later disclaim the order. A

must somehow generate a sequence of bits (a digital signature) which will convince B (and if need be a judge) that A authorized the order. It must be easy for B to validate the digital signature, but impossible for him (or anyone other than A) to generate it (to prevent charges that B was dabbling in the market illegally with A's money).

There are digital signature schemes which do not involve public key cryptosystems but it will be convenient notationally to let A sign message m by computing the signature, $D_A(m)$. Checking a signature will then be done by computing $m = E_A(D_A(m))$. If $E_A(D_A(m))$ produces an illegible message (random bits) then the signature is rejected as invalid. This notation is somewhat misleading because the actual method of generating and validating signatures can be very different from this model; it is retained because it is widely known and because we will not discuss the differences among different digital signature methods, only their common properties.

Digital signature protocols are naturally divided into three parts: a method of signing messages used by A, a method for authenticating a signature used by B, and a method for resolving disputes, used by the judge. It is important to note that two protocols that differ only in the method of resolving disputes are different. Failure to understand this point has led to confusion in the literature [17,20].

We now turn to specific digital signature protocols.

8. A Conventional Signature Protocol

A conventional "signature" protocol relies on the observation that if A and B trust some central authority CA, and if A and B have a secure method of communicating with CA, then A can "sign" a message simply by sending it to CA and relying on CA to adjudicate disputes. This approach is defended by some [17].

This protocol is subject to the weaknesses of centralized key distribution (described earlier).

9. The Basic Digital Signature Protocol

The first public key based digital signature protocol [1], proceeded by having A sign message m by computing $D_A(m)$ and giving it to B as the signed

message. B (or a judge) can compute $E_A(D_A(m)) = m$, thus confirming the correctness of the signed message. A is held responsible for a signed message if and only if it can be verified by applying A's public enciphering key to it.

This protocol can be criticized [16,17,20] on two grounds: First, the public file might have been tampered with. Methods of authenticating the public file, discussed previously under key distribution protocols, solve this problem.

A second criticism is that A has no recourse should his secret deciphering key be compromised and made public. Anyone can sign any message they desire with A's compromised D_A, and A will be held responsible.

It seems clear that A will only agree to this digital signature protocol if he can provide very good physical security for D_A. The loss to A if D_A is compromised can be substantial.

A different method of solving this problem is to alter the dispute resolution protocol so that A is not held responsible for his signature if his secret deciphering key is compromised and made public.

The fact that altering the dispute resolution procedure creates a different protocol has not been fully appreciated, and the preceding two protocols have been confused with each other for this reason. Some criticism of "the" public key digital signature protocol has actually been of this second protocol, and failed to consider the first protocol at all.

If we assume that A knows D_A, then under the second protocol A can make D_A public and effectively disavow the signed message. For this reason, some critics have argued that this protocol is inadequate.

If we assume that A does not know D_A, then he is unable to disavow his signature under this protocol. It is easy to design a system in which this is the case.

The major difference between the second protocol and the first is in the division of risk: in the second protocol B will be left holding the bag if A's signing key is compromised. Clearly, B must be given assurances that this condition is unlikely before he will be willing to use this protocol.

10. The Time-Stamp Protocol

A protocol that would allow A to report loss or theft of D_A and disclaim messages signed after the reported loss yet force A to acknowledge the validity of signatures made before the reported loss must involve the concept of time.

We introduce time into the following protocol by using time-keepers who can digitally time-stamp information given to them. We assume that both A and B have agreed on a set of acceptable time-keepers whose time-stamps will be accepted in dispute resolution.

If A can report that D_A has been lost, then he must report this fact to some agent who will be responsible for answering queries about the current status of D_A, i.e., has it been lost or not. For simplicity, we shall assume this role is played by the central authority, CA. CA will sign messages stating that A's secret deciphering key has not been compromised as of the current time. These signed messages will be called "validity-checks."

In the time stamp protocol, user A signs message m by computing $D_A(m)$ and sending it to B. B then has a time-keeper time stamp the message and obtains a validity-check from CA. If D_A has already been reported lost B rejects the signature, otherwise he accepts.

In dispute resolution, the judge holds that a message has been validly signed if and only if it can be checked by applying A's public enciphering key AND it has been time-stamped prior to any reported loss of D_A.

This protocol provides very good assurance to all parties that they have been dealt with fairly.

The major disadvantage of this protocol, as compared with the basic digital signature protocol, is the requirement that B obtain both a time-stamp and a validity-check, presumably in real time. These requirements force the use of a communications network, which both increases expense and decreases reliability.

If B is willing to obtain the time-stamp and the validity-check after the transaction has been completed, i.e., within a few days, an off-line system can be used. This modified protocol could be used by B either as a fail-soft protocol during communications outages, or as the standard protocol if communication costs are too high.

Off-line operation is cheaper and more reliable, but it exposes B to some risk: A might have recently reported the loss of D_A and B would not know about it. If physical security for secret deciphering keys is good, this risk should be minimal.

11. Witnessed Digital Signatures

If the value of a transaction is high enough, it might be desirable to have a witness physically confirm that A

signed message m. The witness, W, would compute D_W("I, W, physically saw A agree to and sign message m."). It would be necessary for A and B to agree in advance on acceptable witnesses.

The primary advantage of this protocol is that it reduces B's risk. The primary disadvantage is that it forces A to find a (physically present) witness to confirm the transaction.

12. Digital Signature Applications Not Involving Dispute

Not all applications of digital signatures involve contracts between two potentially disputing parties. Digital signatures are also an ideal method of broadcasting authenticated messages from a central source which must be confirmed by many separate recipients, or repeatedly confirmed by the same recipient at different times to insure that the message has not been modified.

One example of such an application is the distribution of network software to individual nodes of a communications network. It would be clearly undesirable for any node to start executing the wrong software. On the other hand, it is very desirable to send updates to the nodes over the network itself. The obvious solution is for updates to be digitally signed by an appropriate network administrator, and for the nodes to check the digital signature prior to executing them.

This example leads naturally to another application of digital signatures in operating system security. A major risk to the security of an operating system is the possibility that the system code that it is executing today is not the same that it was executing yesterday: someone might have put a trap door into the operating system that lets them do anything they please. To guard against this possibility, the operating system could refuse to execute any code in privileged mode unless that code had been properly signed. Carried to its logical conclusion, the operating system would check the digital signature of privileged programs each time they were loaded into central memory If this check were implemented in hardware, it would be impossible for any software changes to subvert it. The machine would be physically incapable of executing code in privileged mode unless that code was signed.

If privileged programs are digitally signed by the programmer who originally wrote them, as well as by various supervisory levels, and if the computer is physically unable to execute unsigned

code in privileged mode, then it is possible to have complete assurance that the privileged programs running on the computer right now have not been modified since they were given there final checkout and signed by the programmer. Of course, this does not necessarily mean that the operating system is secure, but it does eliminate a major class of worries.

13. Conclusions

This paper has briefly described a number of cryptographic protocols. Certainly, these are not the only ones possible; however, they are valuable tools to the system designer: they illustrate what can be achieved and provide feasible solutions to problems of recurring interest.

Further constructive work in this area is very much needed.

14. ACKNOWLEDGEMENTS

It is a great pleasure for the author to acknowledge the pleasant and informative conversations he had with Dov Andelman, Whitfield Diffie, Martin Hellman, Raynold Kahn Loren Kohnfelder, Frank Olken, and Justin Reyneri.

15. BIBLIOGRAPHY

1. Diffie, W., and Hellman, M. New directions in cryptography. IEEE Trans. on Inform. IT-22, 6(Nov. 1976), 644-654.

2. Evans A., Kantrowitz, W., and Weiss, E. A user authentication system not requiring secrecy in the computer. Comm. ACM 17, 8(Aug. 1974), 437-442.

3. Kohnfelder, L.M. Towards a practical public-key cryptosystem. MIT EE Bachelor's thesis.

4. Lipton, S.M., and Matyas, S.M. Making the digital signature legal--and safeguarded. Data Communications (Feb. 1978), 41-52.

5. McEliece, R.J. A public-key cryptosystem based on algebraic coding theory. DSN Progress Report, JPL, (Jan. and Feb. 1978), 42-44.

6. Merkle, R. Secure Communications over Insecure Channels. Comm. ACM 21, 4(Apr. 1978), 294-299.

7. Merkle, R., and Hellman, M. Hiding information and signatures in trapdoor knapsacks. IEEE Trans. on Inform. IT-24, 5(Sept. 1978), 525-530.

8. Rivest, R.L., Shamir, A., and Adleman, L. A method for obtaining digital signatures and public-key cryptosystems. Comm. ACM 21, 2(Feb. 1978), 120-126.

9. Wilkes, M.V., Time-Sharing Computer Systems. Elsevier, New York, 1972.

10. Diffie, W., and Hellman, M.E., Privacy and authentication: an introduction to cryptography, Proceedings of the IEEE Vol. 67, No. 3, Mar. 1979 pp. 397-427.

11. Squires, J. Russ monitor of U.S. phones, Chicago Tribune pp. 123, June 25, 1975.

12. Davis, R. Remedies sought to defeat Soviet eavesdropping on microwave links, Microwave Syst., vol. 8, no. 6, pp. 17-20, June 1978.

13. Merkle, R.C. A certified digital signature, to appear, CACM.

14. Kahn, D. The Codebreakers, New York: Macmillan. 1967.

15. Rabin, M.O., Digitalized signatures, in Foundations of Secure Computation, ed. Demillo, R.A., et. al. pp. 155-166.

16. Saltzer, J. On Digital Signatures, private communication.

17. Popek G.J. and Kline, C.S. Encryption Protocols, Public Key Algorithms, and Digital Signatures in Computer Networks; in Foundations of Secure Computation pp. 133-153.

18. Needham R.M. and Schroeder, M.D. Using Encryption for Authentication in Large Networks of Computers. CACM 21,12 Dec. 1978 pp. 993-999.

19. Merkle, R. Secrecy, authentication, and public key systems. Stanford Elec. Eng. Ph.D. Thesis, ISL SEL 79-017, 1979.

20. Popek, G.J., and Kline, C.S. Encryption and Secure Computer networks. Computing Surveys 11,4 Dec. 1979 pp. 331-356.

21. Simmons, G.J. Symmetric and Asymmetric Encryption. Computing Surveys 11,4 Dec. 1979 pp. 305-330.

22. Lamport, L. Time, clocks, and the ordering of events in a distributed system. CACM 21,7 Jul 1978 pp. 558-565.

23. Ehrsam, W.F., Matyas, S.M., Meyer, C.H., and Tuchman, W.L. A cryptographic key management scheme for implementing the data encryption standard. IBM Sys. Jour. 17,2 1978 pp. 106-125.

24. Lamport, L., Constructing digital signatures from a one way function. SRI Intl. CSL - 98

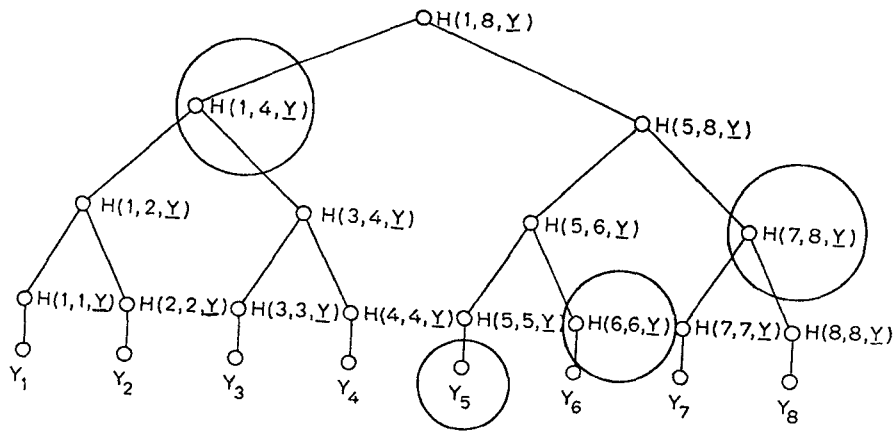

FIG. 1