CHAPTER NINE

Digital Signatures[1]

With the quickened pace of business today, and the large distances which are frequently involved, the time required to obtain a signed agreement may undesirably delay a project. The use of an electronic (or digital) signature may remove this inconvenience. If today's paper-based business transactions were to be implemented exclusively via an electronic communications network, the system would have to rely on signed messages or digital signatures.

The signature would have to be such that the receiver could prove to an impartial third party (a court, judge, or referree before whom the parties had agreed to submit for resolution any issue or dispute)[2] that the contents of the message were genuine and that it originated with the sender. The signature would also have to be such that the sender could not later disavow messages, nor could the receiver forge messages or signatures for self-serving purposes.

A signature is a function of (1) the message, transaction, or document to be signed, (2) secret information known to the sender, and possibly, (3) public information known to all parties. The signature may either be a special bit pattern appended to the data, or it may be an integral part of the cryptographically transformed data.

Although digital and written signatures can serve the same purposes, there are obvious physical differences. It should be understood that the several branches of law pertaining to signatures assume a paper-based system as the medium for transacting business. Thus before describing how digital signatures can be obtained, the legal significance of written signatures is discussed.

SIGNIFICANCE OF SIGNATURES[3]

The legal significance of signatures and the use of writings bearing such signatures must be viewed from a perspective which encompasses several branches of the law, including but not limited to the Statute of Frauds, the Law of

[1] This is a technical discussion of an approach to solving a legal problem. The material contained in this chapter does not constitute advice, and those who intend to implement any of the concepts included herein should first consult their legal counsel.

[2] Hereafter, this impartial third party shall be referred to as the referee.

[3] ©1978 McGraw-Hill, Inc. Reprinted in part from *Data Communications,* February 1978 [1].

Acknowledgments, the Law of Agency, and the Uniform Commercial Code (UCC). The need for a device or process which may satisfy the requirements of such branches of law in the modern context of electronic (or paperless) signatures can be appreciated by understanding some key areas of the law affecting signatures.

The history of the Statute of Frauds [2] begins in 1677 when "An Act for Prevention of Frauds and Perjuries" was enacted in England. The need for such an act resulted from the peculiar rules of evidence used by English courts during the 17th Century. For example, two parties (A and B) might enter into an oral agreement for the sale of land. It was possible for A to sue B, alleging that B orally agreed to sell the land for a certain amount. In fact, there may never have been any agreement at all, or the price may have been much greater. Under English law, B could not testify in his own behalf. Lawsuits of this kind were frequently tried with professional witnesses, testimony of friends of the parties, and the like. Since perjury was commonplace, the defendant in such cases was at a distinct disadvantage. Suppose that party C testified that he heard B agree to the sale. How could B bring forth a witness who could testify that he did not hear the agreement? The difficulty is that of proving a negative condition.

The difficulty was finally overcome by requiring written evidence that contracts were actually entered into. Specifically, the Statute of Frauds was designed to prevent fraud by excluding from consideration by the courts legal actions on certain contracts, unless there was written evidence of the agreement signed by the party to be charged or his duly authorized agent.

Law of Acknowledgments

Certain documents require acknowledgment or proof of the identity of the person who signs the document, and proof that it was signed on the stated date. This acknowledgment or proof is necessary to prevent the person who signed the document from claiming later that the signature is not genuine. Moreover, certain transactions require that the signature be witnessed by one or more persons. Such transactions may vary according to the law of the jurisdiction in which the document was executed.

Acknowledgment or proof of signature upon a legal document or instrument may normally be made before a judge, an official examiner of title, an official referee, or a notary public. Essentially, the form of an acknowledgment consists of the following:

On the_____day of_____, 19__, before me personally appeared (John Doe) to me known and known by me to be (John Doe) who placed his hand upon said document in my presence and acknowledged same to be his signature.

_____

Notary Public

Such acknowledgments together with the signed document are usually recorded in an official registry, like an office of the county clerk or secretary of state.

Law of Agency

The principles of agency law [2] are essential for the conduct of business transactions. A corporation, as a legal entity, can function only through its agents. The law of partnership is to a large degree a special application of agency principles to that particular form of business organization. Agency may be defined as follows:

> Agency is the fiduciary relation (involving a confidence or trust) which results from the manifestation of consent by one person to another that the other shall act on his behalf and subject to his control, and consent by the other so to act. [Restatement of the Law, Agency (2d), p. 7, Sec. 1 (1)]

As a general rule, no particular formalities are required to create an agency relationship. The appointment may be either written or oral, and the relationship may be either expressed or implied. There are two situations in which formalities are required: (1) with a power of attorney, where a formally acknowledged instrument is used for conferring authority upon the agent; and (2) in a few states where it is required that the act which confers authority to perform a certain act must possess the same formalities as the act to be performed. For example, authority to sign a contract which is required to be in writing must itself be granted by a written instrument.

Generally, the law of agency applies to contracts or commercial paper. A principal (the person from whom an agent's authority derives) is bound by the duly authorized acts of his agent. However, if the agent does not possess the requisite authority (express, implied, or apparent), the principal in most instances will not be bound. An agent who fails to bind his principal to an agreement because of the agent's failure to name the principal, or due to lack of the agent's authority, will usually be personally liable to third parties. Thus the correct way for an agent to execute a contract or instrument is to affix the name of his principal followed by his own signature and the capacity in which it is made: "P" Principal, by "A", as Agent.

Uniform Commercial Code

The Uniform Commercial Code (UCC) [3] is a comprehensive modernization and compilation of the various statutes relating to commercial transactions. Its primary objective is to provide uniformity of commercial law throughout American jurisdictions. It has been adopted in all states except Louisiana. The present articles relating to commercial paper, banking transactions, and investment securities are paper-based.

To accommodate electronic funds transfer systems, a special committee was formed to prepare amendments or supplements to these articles. Although the principles governing the transfer of paper-based stocks and bonds (see Article 8 of reference 3, for example) can generally be made applicable to the paperless variety, many technical and mechanical changes are needed to apply those principles to securities without certificates.

According to the current (1972) version of the UCC,

> "Signed" includes any symbol executed or adopted by a party with present intention to authenticate a writing. [UCC: Sec. 1-201 (39)]

and, in case of commercial paper,

> A signature is made by use of any name, including any trade or assumed name, upon an instrument, or by any word or mark used in lieu of a written signature. [UCC: Sec. 3-401 (2)]

The inclusion of the word authenticate in the definition of signed clearly indicates that a complete handwritten signature is not necessary. This authentication may be printed, typed, stamped, or written; it may be initials or thumbprint. It may be on any part of the document, and in certain cases may be found in a billhead or letterhead. No catalog of possible authentications can be complete, and courts must use common sense and commercial experience in passing upon such matters. The question is always whether the symbol was executed or adopted by the party with the intention at that time of authenticating the writing.

A signature may be made by an agent or other representative, and his authority to make such signature may be established according to the Law of Agency. No particular form of appointment is necessary to establish such authority. However, such a signature may be unauthorized if made by an agent who exceeds his actual or apparent authority. An unauthorized signature is one made without actual, implied, or apparent authority, and includes those made by forgers, imposters, and fictitious payees.

The law of commercial paper also recognizes the principle that the drawer—the one who creates a negotiable instrument (a draft, check, note, or certificate of deposit)—has voluntarily entered into relationships beyond his control with subsequent holders of the instrument. The law imposes on the drawer the responsibility to assure that his own negligence does not contribute to the possibility of material alteration of the instrument later in the chain of transfer.

Contributory Negligence

Any person who by his own negligence substantially contributes to a material alteration of the instrument, or to the making of an unauthorized signature, is precluded from asserting the defense of alteration or of lack of authority against anyone who has accepted the instrument in accordance with reasonable commercial standards. An example of such negligence is the situation where space is left in the body of the instrument, such as $ 500, allowing the value to be changed to $2,500, or allowing the words " five hundred" to be changed to "twenty-five hundred." It also covers the most obvious case where a drawer makes use of a signature stamp or other automatic signing device and is negligent in controlling access to it.

In banking transactions, verification of signatures is a necessary part of the procedure known as the process of posting. Completion of this procedure helps to determine when an item is finally paid in favor of an innocent

holder. Posting involves two basic elements: a decision to pay, and some recording of the payment. In certain instances, the recording may actually precede the decision to pay. That is, provisional debits may be entered, and the decision on the authenticity of the signature may be made at a later time.

As incorporated in the UCC, the concept of finality of payment [Sec. 3–418] states that a drawee (the person on whom a bill of exchange is drawn) cannot recover funds paid to a bona fide holder of a draft or check bearing a forged signature of the drawer (one who draws a bill of exchange, or order for payment). This is known as the rule of *Price v. Neal* [3 Burr. 1354 (1762)]. The rule, as enunciated by Lord Mansfield in 1762, imposed upon the drawee the duty to be satisfied that "the bill drawn upon him was in the drawer's hand" [*Price v. Neal*, 3 Burr. 1354, at 1357] before he accepted or paid it, but that it was not the duty of the good-faith holder to inquire into it.

Many banks today rarely review the signature on a small check for its authenticity. Only in cases of stop-payment orders and reports of lost or stolen checks do banks interrupt their otherwise mechanized routines involving such instruments. Generally, losses incurred as a result of forged drawers' signatures are small enough to be absorbed as a cost of operation.

OBTAINING DIGITAL SIGNATURES

For a digital signature procedure to work, there must be enough information available for message and signature validation and yet insufficient information to permit forgery of either message or signature. While a receiver could validate messages and signatures with the same information (algorithm and parameters) used by the sender to create signatures, this could also permit forgery. Therefore, the same information is never sufficient for both signature generation and validation.

Using a data communication system, a sender A may transmit signed messages to a receiver B under a defined procedure which requires that certain information be held by both parties. A must have information that allows it to generate a signature for each message transmitted to B. And B must have information that allows it to validate messages and signatures received from A. The procedure can be extended to permit two-way communication by providing B with signature-generation information similar to that held by A, and by providing A with information that allows it to validate messages and signatures received from B.

If A is concerned that B may later disavow the receipt of messages, A can require that messages be certified. *Message certification* means that the receiver provides some proof to the sender that the message was received. For example, if A sends message M to B, then B could send back to A the signed message "B received M from A," with message M repeated in its entirety.

An initial written and hand-signed agreement is normally required between sender and receiver, regardless of the method used in implementing digital signatures (see Legalizing Digital Signatures). Such an agreement should contain a complete description of the digital signature procedure to be used, including either a list of bit patterns that may be required for validating

signatures or the name and location of a registry where the bit patterns are recorded.

There are several cryptographic techniques for generating digital signatures using both conventional and public-key cryptographic algorithms.[4] Digital signatures obtained with a public-key algorithm are referred to here as *universal, general,* or *true* signatures. This is because such signatures can be validated by *anyone* having access to the public-key (or validation parameter) of the sender. To insure that the public-key is genuine and cannot be changed, each sender must publish or record his public-key in a designated registry.

Universal signatures can also be obtained with a conventional algorithm, but the protocols are more involved. As with a public-key algorithm, the sender must share certain nonsecret validation information in advance with the receiver, and this information must also be published or recorded in a designated registry. However, the protocol may also involve certain other secret and/or nonsecret information sent to the receiver at the time the message is validated. However, in no case does the sender share all his secret signature generation information with other communicants or parties.

Arbitrated signatures are another way to obtain digital signatures with a conventional algorithm. An arbitrated signature is validated by a trusted arbiter at the time the message is communicated between parties. To permit validation, each user shares his secret signature-generation information with the arbiter.

Finally, digital signatures can be obtained with a conventional algorithm if the algorithm is first used to obtain the properties of public and private keys. The digital signatures are then created in the same manner as if they were created with a public-key algorithm.

UNIVERSAL SIGNATURES

When universal signatures are involved, the receiver can independently validate each message and signature. A referee is called upon only to settle disputes.

Figure 9-1 illustrates the general approach for obtaining universal signatures. To prevent signatures from being forged, and to associate them uniquely with their senders, each communicant has certain nonshared (secret) information that is used to generate the signature. Usually a signature and message are validated with nonsecret validation information which is included within the initial written agreement or recorded in a designated registry and is available to the parties in advance. But it may also involve certain secret and/or nonsecret information provided to the receiver at the time the message is validated.

[4] A *conventional* cryptographic algorithm is one for which the enciphering and deciphering keys are either identical or are such that each can easily be computed from the other. In contrast, a *public-key* algorithm is one for which there are public and private keys (normally the public key is used for enciphering and the private key for deciphering), and knowledge of the public key does not permit the private key to be computed [4]. See also Cryptographic Algorithms, Chapter 2.

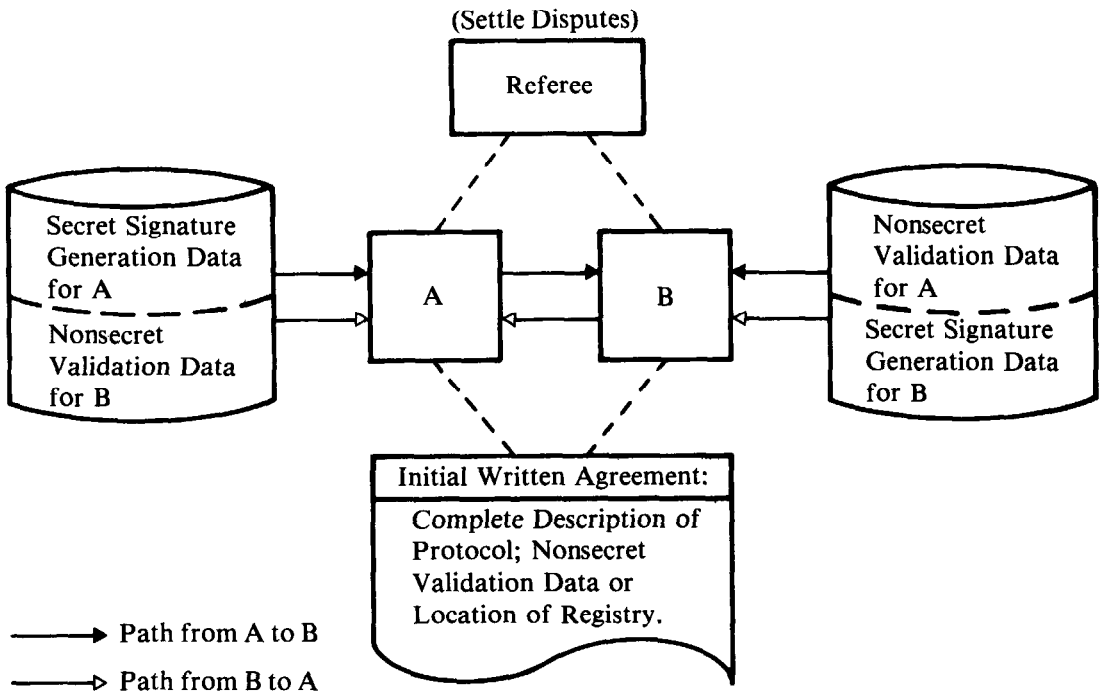


Figure 9-1. A General Approach for Exchanging Universal Signatures

An Approach Using Public-Key Algorithms

Recall that a cryptographic algorithm is composed of enciphering (E) and deciphering (D) procedures which usually are identical or simply consist of the same steps performed in reverse order, but which can be dissimilar.[5] The keys selected by the user consist of sequences of numbers or characters. An enciphering key (Ke) is used to encipher plaintext (X) into ciphertext (Y), as in Equation 9-1, and a deciphering key (Kd) is used to decipher ciphertext (Y) into plaintext (X), as in Equation 9-2.

$$E_{Ke}(X) = Y \tag{9-1}$$

$$D_{Kd}(E_{Ke}(X)) = D_{Kd}(Y) = X \tag{9-2}$$

If E and D are made public, as the present discussion assumes, cryptographic security completely depends on protecting the secret keys.

In a public-key algorithm, the enciphering and deciphering keys, Ke and Kd, are unequal. One key is made public and the other is kept private. However, if the algorithm is to be useful, communicants must be able to compute a public and private pair of keys efficiently, whereas knowledge of the public key alone must not permit the private key to be computed efficiently.

[5] Conventional and public-key algorithms have already been defined and discussed in Chapter 2. However, these subjects are introduced here in a slightly different way.

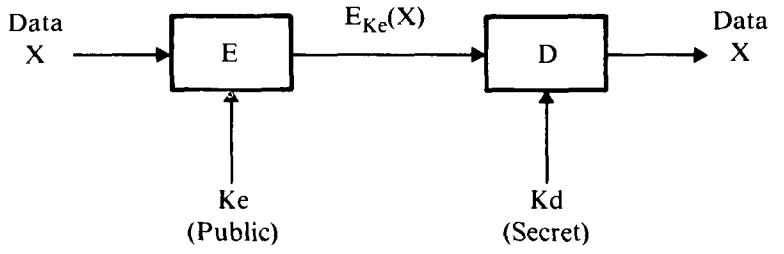


Figure 9-2. Public-Key Cryptographic System Used for Privacy Only

If Ke can be made public without compromising the secrecy of Kd, then the public-key algorithm can be used for private data communications (Figure 9-2). In that case, anyone can encipher data by using the receiver's public enciphering key, but only the authorized receiver can decipher the data through possession of the secret deciphering key.

If Kd can be made public without compromising the secrecy of Ke, then the public-key algorithm can be used to obtain digital signatures, or signed messages (Figure 9-3). In that case, anyone can decipher data using the sender's public deciphering key and thereby prove the origin of the data, but only the authorized sender can encipher the data through possession of the secret enciphering key.

By inserting prearranged information in all messages (such as sender ID, receiver ID, and message sequence number), the messages can be checked to determine if they are genuine. However, because the data are available to anyone having the public deciphering key, privacy is not attained.

If it is also the case that encipherment followed by decipherment, and decipherment followed by encipherment, produce the original plaintext, that is,

$$D_{Kd}(E_{Ke}(X)) = E_{Ke}(D_{Kd}(X)) = X; \quad \text{for all } X \tag{9-3}$$

then the public-key algorithm can be used for both private data communi-

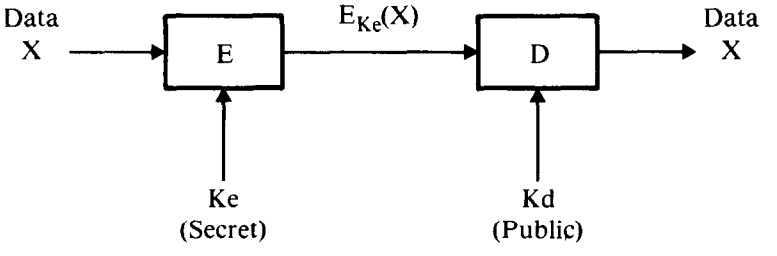
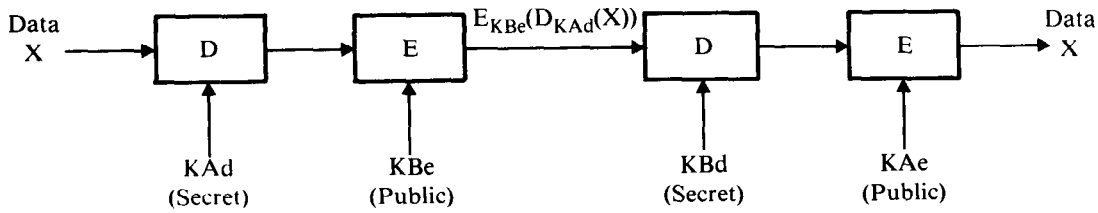


Figure 9-3. Public-Key Cryptographic System Used for Digital Signatures Only



KAe and KAd are enciphering and deciphering keys of the sender (A).
KBe and KBd are enciphering and deciphering keys of the receiver (B).

Figure 9-4. Public-Key Cryptographic System Used for Both Privacy and Digital Signatures

cations and digital signatures (Figure 9-4).[6] A message is signed by deciphering it with the sender's (A's) secret deciphering key (KAd), and privacy is obtained by enciphering the result with the receiver's (B's) public enciphering key (KBe).

However, it must be emphasized that effective data security demands that the correct public key be used; otherwise, the system is exposed to attack. For example, if A can be tricked into using C's instead of B's public key, then C can both decipher the secret communications sent from A to B, and transmit messages to A pretending to be B. Thus key secrecy and key integrity are two distinct, but very important, attributes of cryptographic keys. With a public-key algorithm, the requirement for key secrecy is relaxed for one of the keys, but the requirement for integrity is not.

In order to understand how a public-key algorithm can be used to obtain signed messages, let PA, PB, . . . , PZ denote the public enciphering keys (stored in public files) and SA, SB, . . . , SZ denote the private deciphering keys belonging to the communicants A, B, . . . , Z, as shown in Figure 9-5.

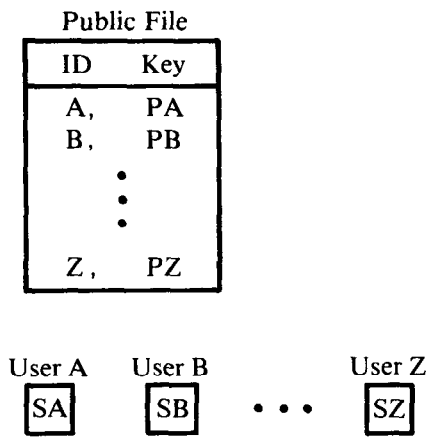


Figure 9-5. NonSecret Enciphering Keys and Secret Deciphering Keys

[6] Merkle and Hellman [5] have shown that a signature capability can be obtained if the relation $E_{Ke}(D_{Kd}(X)) = X$ holds for only a fraction of the set of all messages.

Assume also that the relation in Equation 9-3 is satisfied. Note that for each pair of keys (PA, SA), it follows that $E_{PA}(D_{SA}(M)) = M$ for all M, whereas $E_{PB}(D_{SA}(M)) \neq M$ for almost every M when $PB \neq PA$.

Assume that all communicated messages contain the sender's ID, the receiver's ID, and a message sequence number in addition to whatever other text is sent in the message; that is, let M denote a message whose format is

$$\begin{aligned} M = {} & \langle \text{sender ID} \rangle, \\ & \langle \text{receiver ID} \rangle, \\ & \langle \text{sequence number} \rangle, \\ & \langle \text{data} \rangle \end{aligned} \tag{9-4}$$

where the symbols < and > delimit the elements of M. The text of the message is represented by <data>. (A description of the message format would be part of the prior written agreement between the parties.)

In the general case, M will have to be divided into several blocks and enciphered/deciphered separately using methods of chained block encryption (see Block Ciphers and Stream Ciphers, Chapter 2). However, to simplify the discussion, assume that M contains only one block.

Prior to the commencement of the protocol, each communicant reads the public file and obtains the public key of each other communicant. In the present example, A would obtain PB and B would obtain PA from the public file. (Although keys stored in the public file need not be kept secret, they must be protected from accidental or intentional alteration. Methods have been suggested for protecting the integrity of the public-key file [6], but are not discussed here. See also Key Management Considerations–Symmetric Versus Asymmetric Algorithms, Chapter 11).

Let Mi = [A, B, i, T] denote the ith message communicated from A to B. To sign the message using the described protocol, A deciphers Mi with the secret key SA. If private data communications are also required, the result, $D_{SA}(Mi)$, is enciphered with B's public key, PB, and the resulting quantity, $E_{PB}(D_{SA}(Mi))$, is sent to B; otherwise $D_{SA}(Mi)$ is sent to B.

If $E_{PB}(D_{SA}(Mi))$ is received, B recovers $D_{SA}(Mi)$ by deciphering the received quantity with the secret deciphering key, SB.

$$D_{SB}(E_{PB}(D_{SA}(Mi))) = D_{SA}(Mi)$$

Otherwise, $D_{SA}(Mi)$ is received, in which case decipherment with SB is not required. In any event, B then recovers Mi by enciphering $D_{SA}(Mi)$ with A's public key, PA.

$$E_{PA}(D_{SA}(Mi)) = Mi$$

Once the content of Mi has been verified, quantity $D_{SA}(Mi)$ can be used as proof that Mi originated with A. Because SA is available only to A, only A could produce $D_{SA}(Mi)$ in the first place. It is impossible for another communicant to produce $D_{SA}(Mi')$ for a given Mi′ (unless, of course, SA is compromised).

Let Q denote the concatenation of <sender id>, <receiver id>, and <sequence number>, and let c be the length of Q in bits. Assume that the public-key algorithm is a good generator of pseudo-random numbers. Therefore, if $D_{SA}(Mi)$ is enciphered with any key other than PA, or a corrupted value of $D_{SA}(Mi)$ is enciphered with any key including PA, then the probability of producing a correct value Q is approximately equal to $1/2^c$.

Because $1/2^c$ is a very small number (recall that c is the length of Q in bits), the content of Mi can be authenticated by ensuring that the correct value of Q is recovered when $D_{SA}(Mi)$ is enciphered with PA.[7] Therefore, B concludes that M is genuine and that it originated with A if and only if

1. <sender ID> = A (sender's ID),
2. <receiver ID> = B (receiver's ID)
3. <sequence number> = i (next expected value).

In the example, $D_{SA}(Mi)$ is a function of both the message (Mi) and the sender's secret key (SA), and can be computed only by the sender. Moreover, $D_{SA}(Mi)$ satisfies the criteria for a signed message without the need for a digital signature to be appended to the message. This is because the contents of a recovered message can be validated strictly on the basis of known parameters contained in the data, such as <sender id>, <receiver id>, and <sequence number>.

Schemes for obtaining digital signatures with a public-key algorithm have been invented by Rivest, Shamir, and Adleman [7], Merkel and Hellman [5], and Shamir [8].

An Approach Using Conventional Algorithms

Method One

An approach invented by Diffie and Lamport [4], which is based upon a conventional algorithm such as DES, makes use of a digital signature composed of a list of cryptographic keys.

In order that an n-bit message may be signed, the sender randomly generates in advance 2n 56-bit cryptographic keys

$$k1, K1, k2, K2, \ldots, kn, Kn \tag{9-5}$$

which are kept secret. The receiver is given in advance two sets of corresponding nonsecret 64-bit validation parameters

$$u1, U1, u2, U2, \ldots, un, Un \tag{9-6}$$

[7] If M is divided into blocks and a method of chained block encryption is used, then the last block must contain some data whose value is known to the receiver. This may require that some agreed upon constant, say all zero bits, be included in the last block.

and

$$v1, V1, v2, V2, \ldots, vn, Vn \tag{9-7}$$

where

$$vi = E_{ki}(ui), Vi = E_{Ki}(Ui); \quad i = 1, 2, \ldots, n$$

The validation parameters (Equations 9-6 and 9-7) are also recorded by the sender in an established public registry with recognized and accepted integrity. It is the receiver's responsibility to ensure that the validation parameters received from the sender are genuine (e.g., by comparing them with similar values stored in the public registry).[8]

Later, when message M is sent, the digital signature is generated by selecting k1 or K1 depending on whether the first bit of M is 0 or 1, respectively; selecting k2 or K2 depending on whether the second bit of M is 0 or 1, respectively; and so forth. For example, if M = 0, 1, 1, 0, . . . , the signature would contain the following keys: k1, K2, K3, k4, ...

The receiver validates the digital signature by ensuring that the first 56-bit key in the signature will encipher validation parameter u1 into $E_{k1}(u1)$ if the first bit of M is 0, or that it will encipher U1 into $E_{K1}(U1)$ if the first bit of M is 1; the second 56-bit key in the signature will encipher validation parameter u2 into $E_{k2}(u2)$ if the second bit of M is 0, or it will encipher U2 into $E_{K2}(U2)$ if the second bit of M is 1; and so forth.

For all practical purposes, only the sender, who knows the secret values of ki and Ki (Equation 9-5) and who originally creates vi and Vi from ui and Ui, can disclose a key (to the receiver) that will successfully encipher either ui into vi or Ui into Vi. An opponent would have to discover the value of one of the secret keys (e.g., by using Ui and Vi to solve for Ki), which would not be computationally feasible, or change the value of one of the validation parameters recorded in the registry. For these reasons, and since the procedure is a one-time system (i.e., ki, Ki, vi, and Vi are not reused once ki or Ki has been sent to the receiver), the receiver is unable to forge a single bit in the message.

The above approach demonstrates that digital signatures can be obtained with conventional algorithms and is discussed because it is simple and easy to understand. An obvious disadvantage is that a separate key must be included in the signature for each bit in the message. This could result in very large signatures depending upon the size of the message to be signed. For example, if the DES algorithm were used, the approach would result in a 56-fold data expansion. However, if data compression techniques are used, smaller signatures are possible.

[8] Alternatively, only one validation parameter (a master validation parameter) is stored in the public registry, and this one value is used to validate the entire set of validation parameters received from the sender. The master validation parameter could be published in one or more major newspapers. In such an approach, the master validation parameter would be a complex function of the set of validation parameters (see Compressed Encoding Function). However, such protocols are omitted from the discussion.

Compressed Encoding Function

Let *compressed encoding* (CE) be a function that maps variable length messages of t bits to fixed length bit patterns of m bits, where m and CE are such that it is computationally infeasible to find two different messages, M and M′, for which CE(M) = CE(M′) (Figure 9-6). For a specific M, it ordinarily requires on the order of $2^m/2 = 2^{m-1}$ trials to find an $M' \neq M$ such that CE(M′) = CE(M). However, by trading off time for memory, the number of trials can be reduced with the aid of a precomputed table (see reference 9 and Appendix B). (However, the time to compute such a table may be considerable.) Suppose, for example, the CEs of $2^{m/2}$ messages have been computed and stored in a table. Since the total number of CEs (2^m) is much greater than $2^{m/2}$, the probability that an intercepted CE is included in the table is about $1/2^{m/2}$. In that case, only about $2^{m/2}$ CEs must be intercepted before a match with a table entry occurs. In other words, to find two different messages, M′ and M, such that CE(M′) = CE(M), requires $2^{m/2}$ trials instead of $2^m/2$. If m = 64, the values for 2^{32} and 2^{63} are

$$2^{32} = 4{,}294{,}967{,}296$$

$$2^{63} = 9{,}223{,}372{,}036{,}854{,}775{,}808$$

Thus, the opponent gains an enormous advantage: a computationally infeasible problem is now reduced to a computationally feasible problem.

For example, a digital signature procedure that uses a CE function with 2^m different CE-values could be defeated if it were possible to compute the compressed encoding function of only twice the square root of 2^m messages.[9]

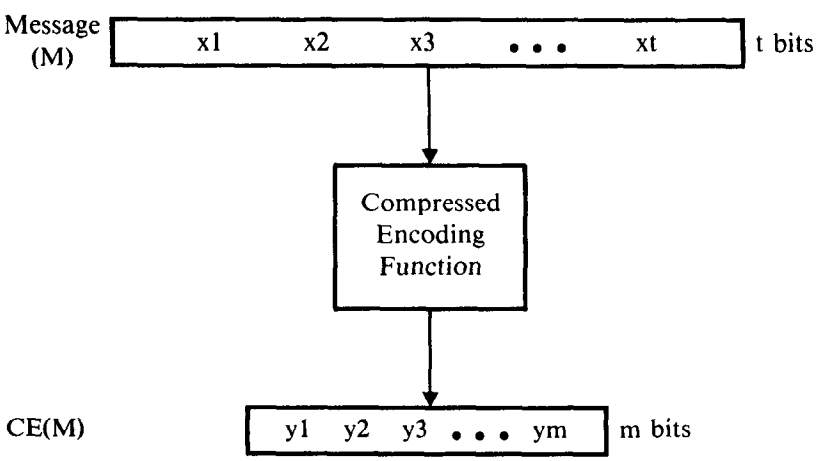


Figure 9-6. The Compressed Encoding Function

[9] If there are 2^m different CE-values, the probability of a match (i.e., any two values being equal) is about 0.5 when only $2^{(m+1)/2}$ CE-values have been precomputed. The attack is described by Yuval [10] in terms of the birthday paradox—a closely related but different problem (see also Appendix B).

Let M1 denote a set of messages that the receiver is willing to accept and let M2 denote a set of messages that the receiver is not willing to accept. The messages in each set could be random perturbations on only two starting messages (i.e., replace words by their synonyms, manipulate the number of blanks between words, and so forth). Generate the compressed encoding for each of these perturbed messages and wait until the same compressed encoding is produced for an acceptable message in M1 and an unacceptable message in M2.

There are many ways in which a suitable CE function could be defined using the DES algorithm. First let M be divided into n 64-bit blocks, X1, X2, . . . , Xn, and define ∇K as a function that transforms n × 64 bits of data, M = (X1, X2, . . . , Xn), into 64 bits of data (U):

$$\nabla K(M) = U$$

where K is a cryptographic key and U is computed as follows:

$$\begin{aligned} E_K(X1) &= Y1 \\ E_K(X2 \oplus Y1) &= Y2 \\ &\vdots \quad\quad \vdots \\ E_K(Xn \oplus Yn - 1) &= Yn \\ E_K(X1 \boxplus X2 \boxplus \ldots \boxplus Xn \boxplus Yn) &= U \end{aligned}$$

where $\boxplus$ denotes a hashing operator. For example, $\boxplus$ could be simple modulo 2^k addition, k = 1, 2, . . . , 64, or, in an extreme case, a complex encryption function. Now let K1 and K2 (K1 ≠ K2) be two nonsecret keys, and define CE(M) as the 128-bit compressed encoding function

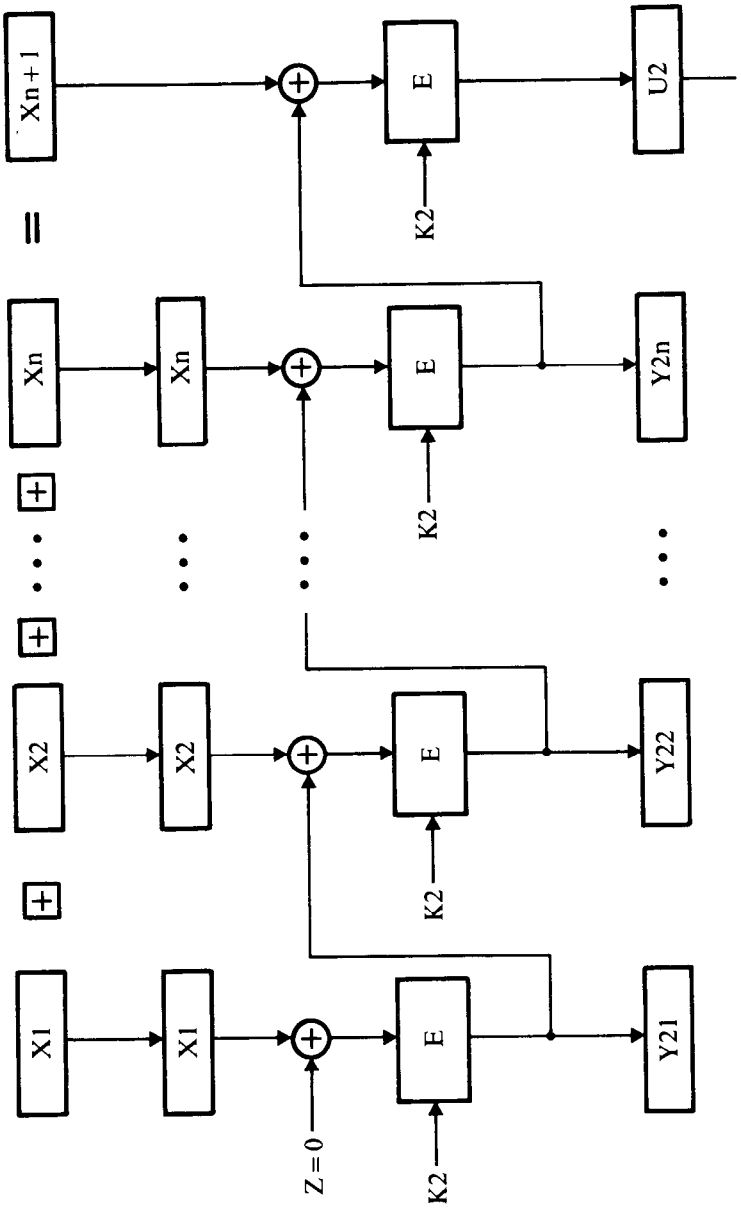
$$CE(M) = (U1,U2) \tag{9-8}$$

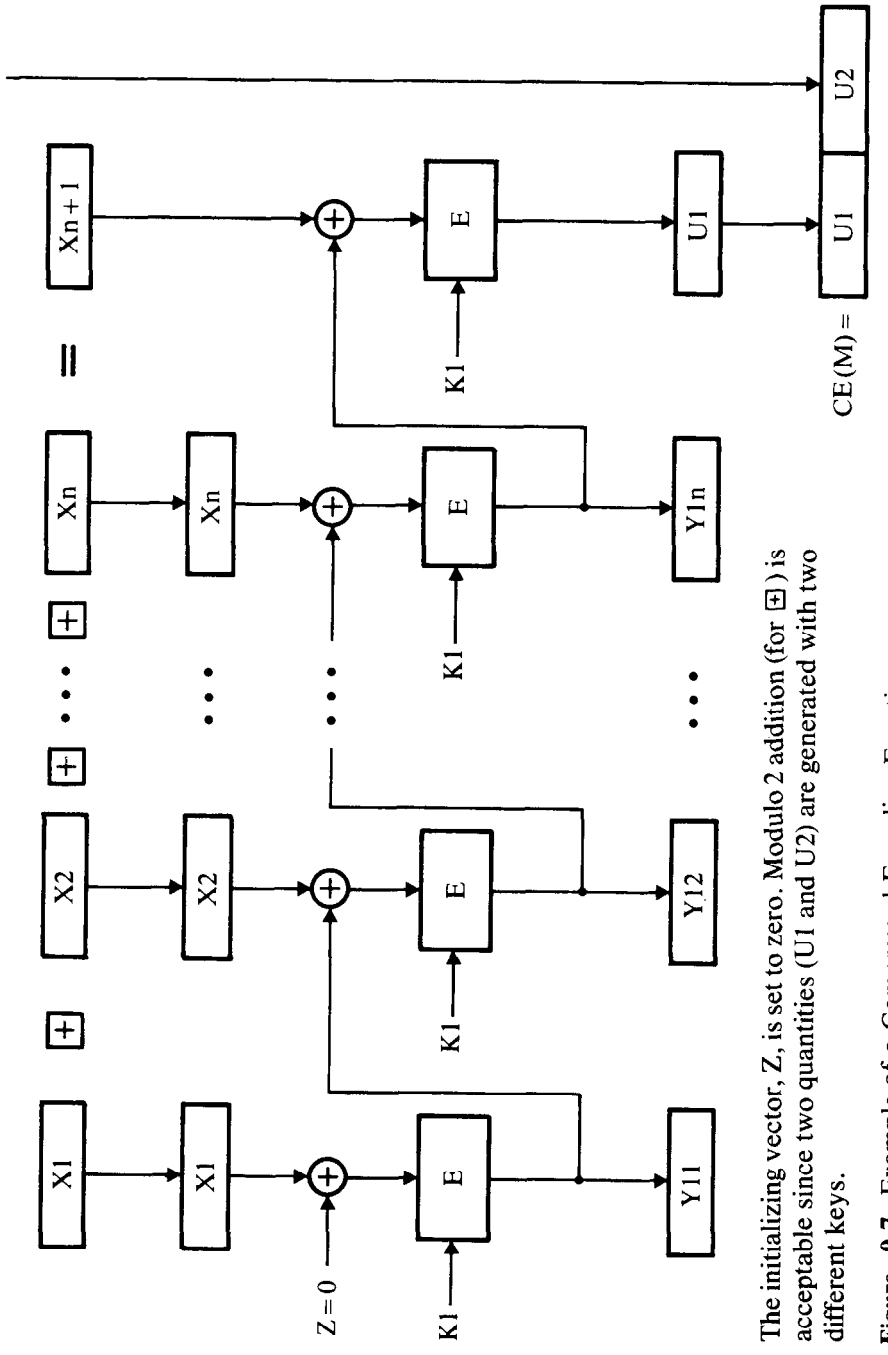
where U1 = ∇K1(M) and U2 = ∇K2(M). The CE function described above is shown in Figure 9-7. The number of bits in the compressed encoding function (Equation 9-8) has thus been adjusted to effectively thwart the time-memory attack suggested by Yuval [10].

Method One—Improved

The number of 56-bit keys comprising the signature is reduced by basing the digital signature on the compressed encoding of M, rather than on M itself. Since CE(M) contains m bits, the resulting signature contains 56m bits, rather than 56t bits. If the number of message bits (t) is much larger than the number of bits in the compressed encoding of the message (m), a significant saving is achieved. If the CE function is computed using Equation 9-8, there are 128 keys, or 56·128 = 7168 bits in the signature (Figure 9-8).

The signature is validated against the 128-bit compressed encoding of the received message via the technique described in method one above. However, the size of the signature can be reduced from 128 to 64 keys if the compressed


Xn+1
E
U2
K2
Xn
Xn
E
Y2n
K2
X2
X2
E
Y22
K2
X1
X1
E
Y21
Z=0
K2



The initializing vector, Z, is set to zero. Modulo 2 addition (for ⊞) is acceptable since two quantities (U1 and U2) are generated with two different keys.

Figure 9-7. Example of a Compressed Encoding Function

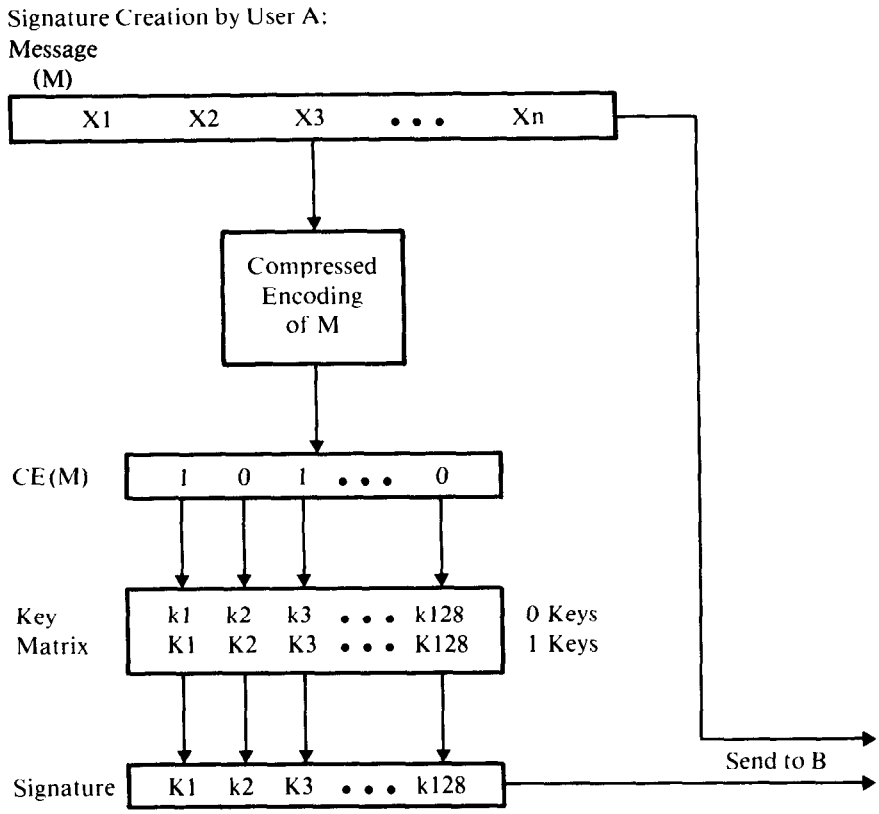


Figure 9-8. A Digital Signature Composed of a List of Cryptographic Keys (Signature Creation)

encoding is divided into 64 2-bit segments instead of 128 1-bit segments, and if the keys k1, k2, . . . , k128, K1, K2, . . . , K128, are arranged in a 4 X 64 table instead of a 2 X 128 table. The rows of the table are denoted 00, 01, 10, and 11, and a key is selected from the appropriate row depending on whether the corresponding bits in CE(M) are 00, 01, 10, or 11.

The same principle could be used to obtain a signature of 32 keys with a 16 X 32 key table, a signature of 16 keys with a 256 X 16 key table, and so on. However, after only a few such reductions, the key table becomes too large to be useful.

Method Two

In a different approach invented by Rabin [11], the digital signature is composed of a list of cryptographic quantities that are formed by enciphering the compressed encoding of a message with a list of randomly selected cryptographic keys. The signature, or list of cryptographic quantities, is validated in a probabilistic fashion by requiring the sender to reveal part of the secret keys that were originally used to produce the cryptographic quantities. Because the particular subset of keys (selected by the receiver) is not known

to the sender ahead of time, the properties essential to digital signatures are obtained.

The sender randomly generates 2r cryptographic keys

$$k1, k2, \ldots, k2r-1, k2r \tag{9-9}$$

which are kept secret. (The value r is determined by security considerations, and is explained later.) The receiver is given, in advance, the corresponding validation parameters

$$u1, u2 \ldots, u2r-1, u2r$$

and

$$E_{k1}(u1), E_{k2}(u2), \ldots, E_{k2r-1}(u2r-1), E_{k2r}(u2r)$$

which have also been recorded by the sender in an established registry with recognized and accepted integrity.

The digital signature for message M is formed (Figure 9-9) by enciphering the compressed encoding of M with each cryptographic key in the list denoted by Equation 9-9:

$$E_{k1}(CE(M)), E_{k2}(CE(M)), \ldots, E_{k2r}(CE(M)).$$

(If CE(M) is computed via Equation 9-8, then $E_{Ki}(CE(M)) = [E_{Ki}(u1), E_{Ki}(u2)]$, and each of the entries in the above list is composed of two 64-bit blocks of ciphertext.) The message and signature are then sent to the receiver.

To permit the signature to be validated, the sender discloses to the receiver exactly half (r) of the secret keys (k1, k2, . . . , k2r) used in forming the signature. The remaining r keys are kept secret in order to prevent the receiver from forging signatures. The keys to be disclosed (Figure 9-10) are selected as follows. The receiver randomly generates a vector of numbers, in which there are exactly r ones and r zeros, and sends a copy to the sender. A 1 in position i of the vector signifies that the sender is directed to forward the ith key to the receiver; a 0 signifies that the ith key has not been selected. Once the r keys have been received, the signature is validated by ensuring that ki1 enciphers ui1 into $E_{ki1}(ui1)$, and ki1 enciphers CE(M) into $E_{ki1}(CE(M))$; ki2 enciphers ui2 into $E_{ki2}(ui2)$, and ki2 enciphers CE(M) into $E_{ki2}(CE(M))$; and so forth.

The sender challenges a message M alleged by the receiver to be signed with $E_{k1}(CE(M)), E_{k2}(CE(M)), \ldots, E_{k2r}(CE(M))$ by producing in front of a referee the keys k1, k2, . . . , k2r. These keys and the alleged signature are then validated as before using the corresponding validation parameters obtained from the designated public registry. If r + 1 or more of the elements in the signature are correct, then the receiver is upheld and it is assumed that the sender is attempting to disavow a message he actually sent. On the other hand, if r or fewer of the elements in the signature are correct, then the sender is upheld and it is assumed that the receiver is attempting to forge a message and signature.

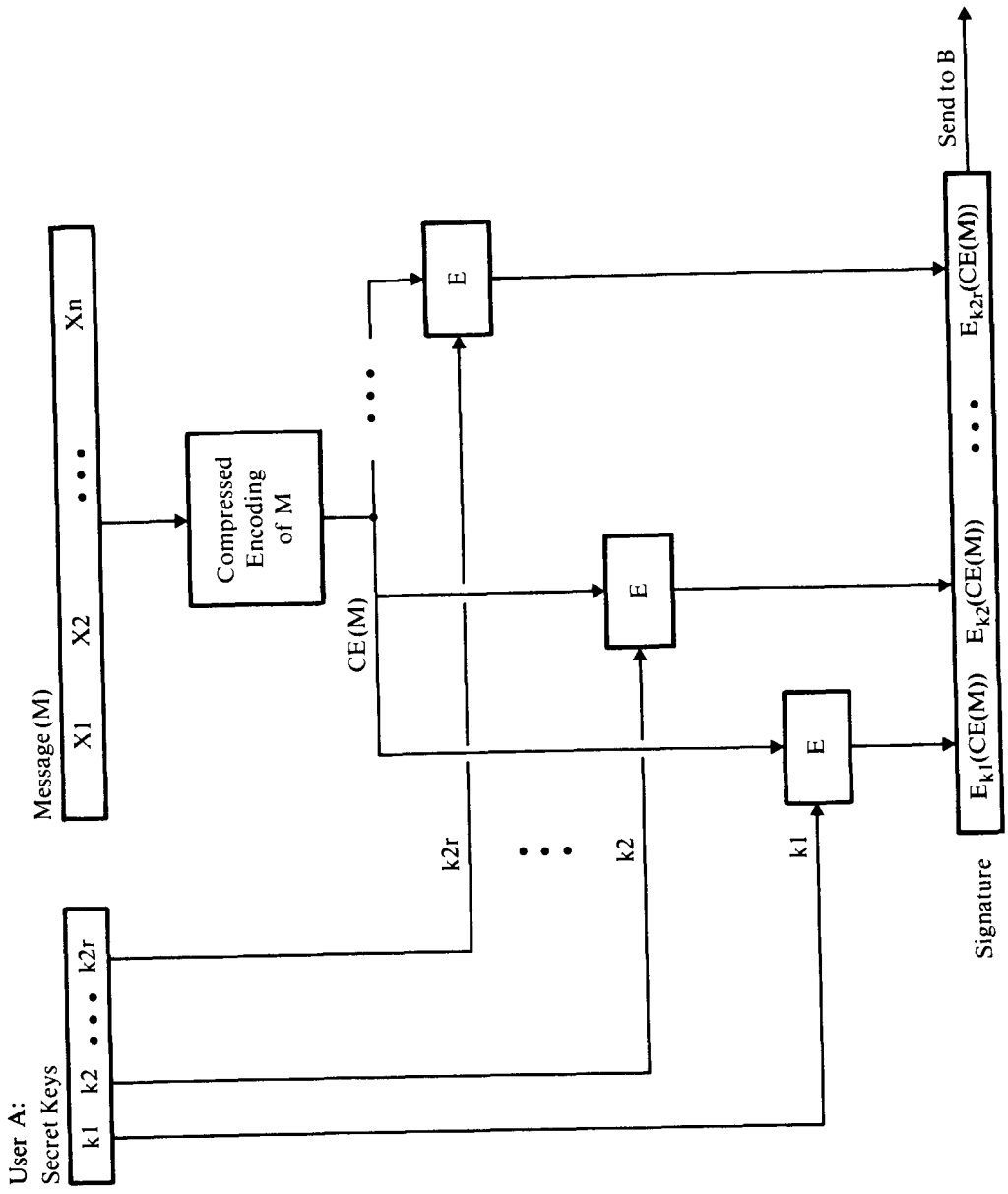


Figure 9-9. Signature Creation

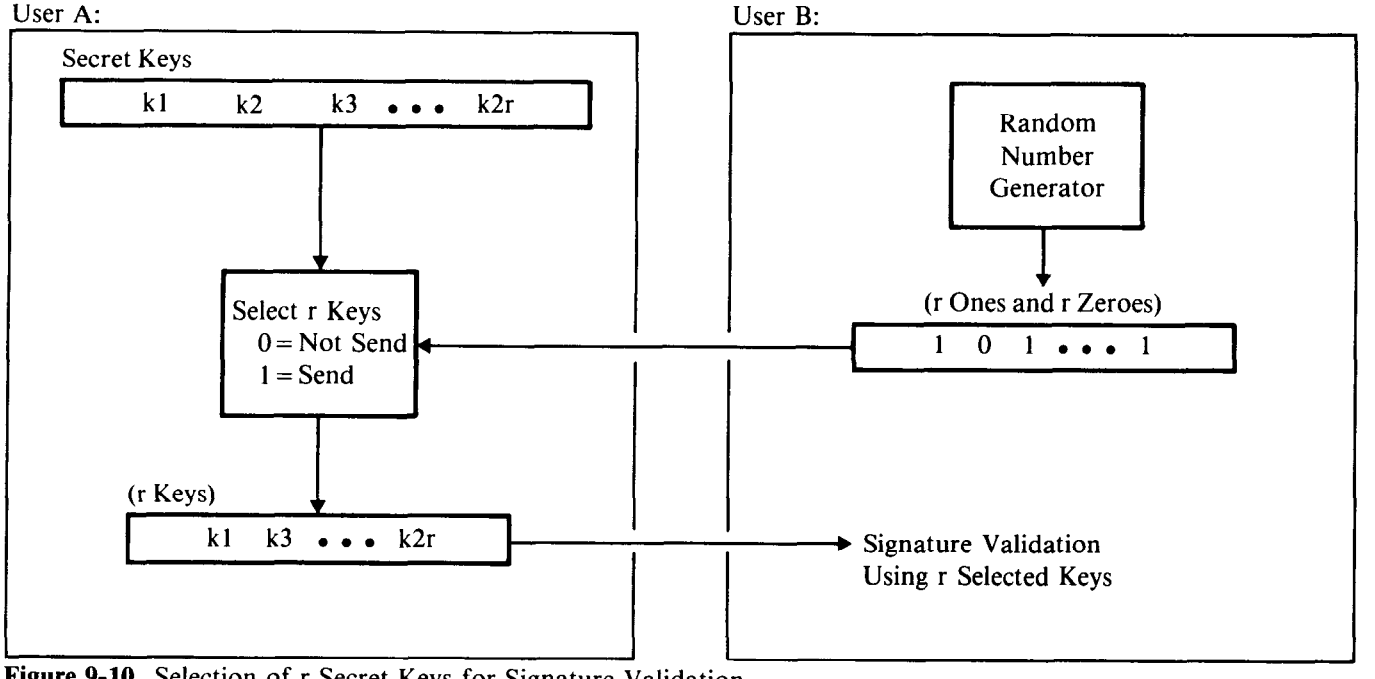


Figure 9-10. Selection of r Secret Keys for Signature Validation

In order for the sender to disavow successfully a message and signature that might also be validated by the receiver, there must be exactly r correct entries in the signature, and the receiver must request and validate the signature using precisely these r entries. But the probability of such an event is given by

$$P = 1/\binom{2r}{r}$$

If $2r = 36$, then the probability is about $1/10^{10}$.

Method Three—Matrix Method

In yet another approach invented by the authors [12], the digital signature is a list of 31 cryptographic keys that are selected from a 31 X 31 matrix of secret keys (Figure 9-11) based on the value of the compressed encoding of the message to be signed. A different matrix of keys is required for each signed message.

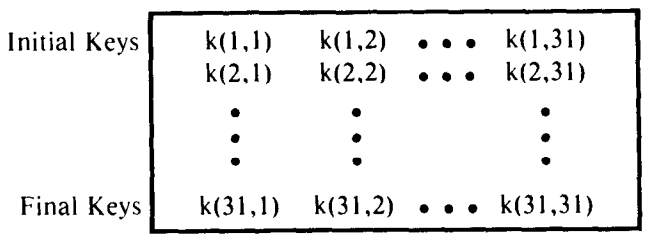


Figure 9-11. 31 × 31 Matrix of Secret Keys used by the Sender to Form a Signature

Row 1 of the matrix contains 31 *initial keys*, denoted k(1, 1), k(1, 2), . . . , k(1, 31), which are produced via a standard generator of pseudo-random numbers. For example, the initial keys for the nth message could be produced with the DES algorithm using a single secret key-encrypting key K, via the relation

$$E_K(31(n-1)+j) \equiv k(1,j)$$

where

$$j = \text{(matrix) column number, } 1 \leqslant j \leqslant 31$$

$$n = \text{message sequence number}$$

In addition to the key matrix, a 30 X 31 matrix of nonsecret *code words* (Figure 9-12) is produced via a generator of pseudo-random numbers. For the purpose of illustration, it is assumed that the code words for the nth message are produced from a single nonsecret seed key U, via the relation

u(1,1)	u(1,2)	• • •	u(1,31)
u(2,1)	u(2,2)	• • •	u(2,31)
•	•		•
•	•		•
•	•		•
u(30,1)	u(30,2)	• • •	u(30,31)

Figure 9-12. 30×31 Matrix of Nonsecret Code Words

$$E_U(31^2(n-1) + 31(i-1) + j) \equiv u(i,j)$$

where

i = (matrix) row number, $1 \leqslant i \leqslant 30$

j = (matrix) column number, $1 \leqslant j \leqslant 31$

n = message sequence number

(The subscript n, which denotes the message sequence number, is omitted from the discussion.)

This method for generating keys and code words has the advantage that the necessary quantities can be created as needed (i.e., they do not have to be saved).

The keys in rows 2 through 31 of the key matrix are obtained via repeated steps of encipherment using the initial keys and the matrix of code words, as follows:

$$E_{K(i,j)}(u(i,j)) \equiv k(i+1,j)$$

where

$$1 \leqslant i \leqslant 30$$

$$1 \leqslant j \leqslant 31$$

For example, k(2, j) is produced by enciphering u(1, j) with k(1, j); k(3, j) is produced by enciphering u(2, j) with k(2, j); and so forth. Thus if the receiver is sent key k(i, j), he can compute k(i + 1, j), k(i + 2, j), . . . , k(31, j), but cannot compute k(1, j), k(2, j), . . . , k(i − 1, j).

The keys in row 31 of the key matrix are called the *final keys*. These final keys, which are prepared in advance by the sender and sent to the receiver, represent the validation pattern. (A protocol must be established whereby the receiver can independently validate these final keys; e.g., by requiring the sender to record them in a designated public registry.) In an actual implementation, a large number of these validation patterns would be needed—one for every message to be signed.[10]

[10] To reduce storage further, the receiver could store the compressed encoding of each validation pattern instead of the validation pattern itself. This modified protocol is not discussed.

The first step in forming the signature is to obtain the compressed encoding of the message M that is to be signed. For the purposes of illustration, assume that the compressed encoding of M, or CE(M), is computed via Equation 9-8; that is, CE(M) = (U1, U2) where U1 and U2 are 64-bit quantities. The compressed encoding of M is used together with 31 nonsecret keys, a1, a2, . . . , a31, (also held by the receiver to permit signature validation) to produce 31 unique code words, b1, b2, . . . , b31, as follows:

$$E_{a1}(U1) \oplus E_{a1}(U2) = b1$$
$$E_{a2}(U1) \oplus E_{a2}(U2) = b2$$
$$\vdots$$
$$E_{a31}(U1) \oplus E_{a31}(U2) = b31$$

The keys, a1, a2, . . . , a31, could be established universal constants that are recorded in a public registry.

The 31 b-values are now sorted into numerical sequence. Since each b-value has a position in the sorted and unsorted sequence, it constitutes an index that can be used to select a key from the key matrix. Its position in the sorted sequence specifies the row, and its position in the unsorted sequence specifies the column.

For example, if the sorted and unsorted b-values are given by

$$\text{Unsorted: } b1, b2, b3, \ldots, b31$$
$$\text{Sorted: } b5, b20, b11, \ldots, b7$$

then the keys in the signature are, respectively,

$$k(5, 1), k(20, 2), k(11, 3), \ldots, k(7, 31)$$

This guarantees that *one and only one key is selected from each row and column.*

As a consequence, one can choose from 31 positions in column 1, 30 positions in column 2, and so on. Altogether, there are 31! or about 2^{112} ways in which the 31 signature keys can be selected from the key matrix. The dimensions of the key matrix have thus been adjusted to eliminate any problem with synonyms (two or more compressed encodings of different messages that result in the same list of signature keys); for example, if the digital signature procedure were attacked using a form of time-memory trade-off [9].

The receiver, except for one additional step, checks the signature by repeating the same steps performed by the sender. First, the compressed encoding of the message is obtained. Then the 31 b-values are computed and sorted into numerical order. This allows the 31 keys in the signature vector to be reinserted into an empty 31 × 31 key matrix. The receiver then uses each signature key to encipher the standard, nonsecret code word to form each lower key in the same column of the matrix, including the final keys that make up the validation pattern. If the final keys (row 31) of the key matrix are equal to the validation pattern previously received from the sender, then

the message and signature are accepted as valid; otherwise, the message and signature are rejected.

If the *rank* of a key is defined as the row number of that key in the key matrix, then in order for the receiver to forge a signature, he must know the value of at least one key in one column of the key matrix whose rank is less than the rank of the corresponding key in the signature. But by the non-invertibility property of the DES algorithm, it is computationally infeasible to compute such a key. Only the sender, who possesses the key-generating key, can produce the full set of keys in the key matrix.

The advantage of this method is that it allows a trade-off to be made between computation time (the time to generate and validate signatures) and signature size. The general approach of computing a list of final keys from a list of signature keys leads to a variety of signature methods. For example, an approach in which both the sender and receiver have enough information to disprove a false claim brought by the other, but do not have enough information to prove a self-initiated claim is described in reference 13.

A requirement of all the methods using conventional algorithms is that they are *one-time systems*. This means that *at least one different validation parameter is required for each message and digital signature that is received.* However, with a public-key algorithm, only one validation parameter (the public-key of the sender) is required, regardless of how many signed messages are received.

ARBITRATED SIGNATURES[11]

The requirement of a one-time system of digital signatures based on a conventional algorithm can be overcome if a protocol of arbitrated signatures is adopted. In a communication system with arbitrated signatures, every signed message prepared by the sender is sent to an arbiter. The arbiter authenticates each message and signature, and communicates the result to the intended receiver. Each communicant's signature-generation information must therefore be shared with the arbiter to allow message and signature validation. Message and/or signature forgery is prevented because the signature-generation information is not shared with other communicants.

Figure 9-13 illustrates an approach for generating arbitrated signatures. In an actual implementation, the arbiter might be a combination of software and hardware located at a communications network node. However, because the arbiter has access to every communicants signature-generation information, a high level of physical security and access control is required.

Arbitrated digital signatures seem best suited to users under a common supervising entity (e.g., brokerage house members of a stock exchange, or users whose transactions occur within a single organization's communication network).

Arbitrated signatures are obtained using message authentication tech-

[11] ©1979 North-Holland Publishing Co. Reprinted in part from *Computer Networks* 3, No. 2, April 1979 [14].

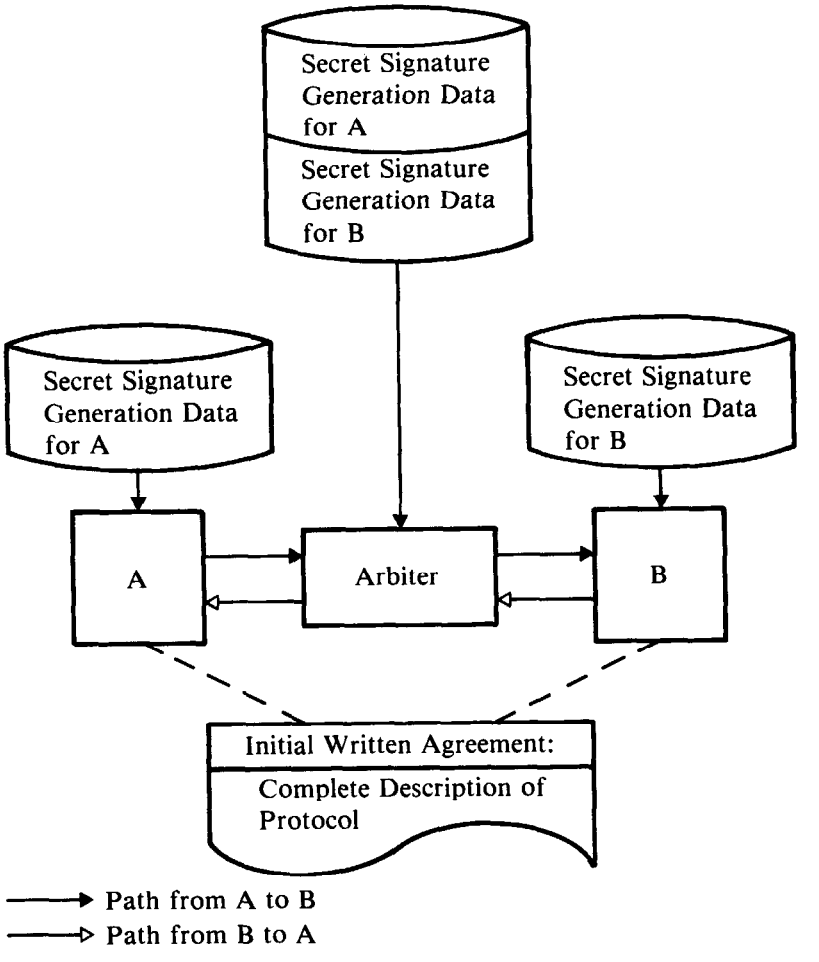


Figure 9-13. Arbitrated Digital Signatures

niques.[12] However, the combination of message authentication and a trusted arbiter (who validates the signature at the time a message is communicated) permits a stronger protocol of digital signatures to be obtained.

Various encryption-based authentication procedures are available that could be used to implement arbitrated signatures. One approach—based on the DES algorithm—is illustrated below.

An Approach Using the DES Algorithm

Suppose each communicant, A, B, . . . , Z, has a secret cryptographic key, KA, KB, . . . , KZ, which is shared with the arbiter (Figure 9-14). With these shared keys it is possible for the arbiter to authenticate messages from each

[12] *Message authentication* is a procedure for authenticating a message's origin, true contents, timeliness, and intended destination. See also Authentication Techniques Using Cryptography, Chapter 8, and Chapters 10 and 11 which treat electronic funds transfer systems.

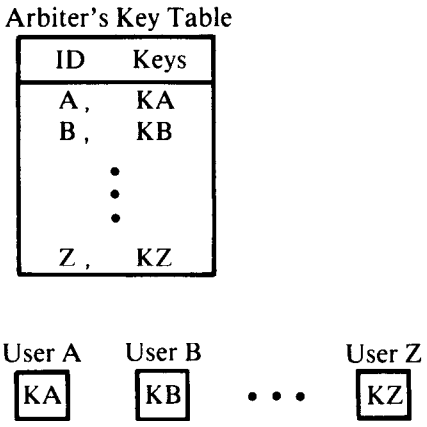


Figure 9-14. Individual Secret Keys Shared With the Arbiter

communicant and for each communicant to authenticate messages from the arbiter. (However, communicants cannot authenticate messages received directly from other communicants.)

In the discussion that follows, let M denote a message whose format is

$$\begin{aligned} M = &\langle\text{sender ID}\rangle, \\ &\langle\text{receiver ID}\rangle, \\ &\langle\text{sequence number}\rangle, \\ &\langle\text{data}\rangle \end{aligned}$$

(see Equation 9-4). Let M be divided into 64-bit blocks of plaintext as follows:

$$M = (X1, X2, \ldots, Xn), \tag{9-10}$$

where Xn could be a short block that is padded to 64 bits. Let X0 be a random 64-bit value that is appended to the front of M and let

$$Xn + 1 = X0 + X1 + \ldots + Xn \pmod{2^{64}}$$

be a block of plaintext appended to the end of M. X0 is used in place of a random initializing vector to mask patterns in the data. X0 can be produced by a generator of pseudo-random numbers. Xn + 1 is a parameter used in the computation of the *authentication code* (AC).

The plaintext X0, X1, . . . , Xn + 1 is enciphered using a form of chained block encryption with ciphertext feedback (Figure 2-17) as follows:

$$\begin{aligned} Y0 &= E_K(X0 \oplus Z) \\ Y1 &= E_K(X1 \oplus Y0) \\ &\vdots \\ Yn+1 &= E_K(Xn + 1 \oplus Yn) \end{aligned} \tag{9-11}$$

where $\oplus$ denotes modulo 2 addition, K is the cryptographic key, and Z (the nonsecret initializing vector) is an established constant, say all zero bits.

Once the message has been enciphered, the following information is sent to the receiver:

$$[Y0, Y, AC]$$

where

$$\begin{aligned} Y0 &= \text{the block of ciphertext corresponding to } X0 \\ Y &= \text{the ciphertext } Y1, Y2, \ldots, Yn \\ AC &= \text{the ciphertext } Yn + 1^{13} \end{aligned}$$

When it is received, the ciphertext $(Y0, Y1, \ldots, Yn + 1)$ is deciphered as follows:

$$\begin{aligned} X0 &= D_K(Y0) \oplus Z \\ X1 &= D_K(Y1) \oplus Y0 \\ &\vdots \\ Xn + 1 &= D_K(Yn + 1) \oplus Yn \end{aligned}$$

where $X0, X1, \ldots, Xn + 1$ denotes the recovered plaintext.

Message authentication is made possible because the recovered value $Xn + 1$ is a complex function of K, and $Y0, Y1, \ldots, Yn + 1$. Therefore, with high probability, decipherment of $Y0, Y1, \ldots, Yn + 1$ with the wrong key, or the corruption of any bits in $Y0, Y1, \ldots, Yn + 1$ will cause the recovered value of $Xn + 1$ to be incorrect. Thus by comparing the recovered value of $Xn + 1$ with $X0 + X1 + \ldots + Xn \pmod{2^{64}}$ for equality, the receiver can determine if the contents of the recovered plaintext $(X0, X1, \ldots, Xn + 1)$ are genuine. And, once the contents of the message have been validated, the parameters, <sender id>, <receiver id>, and <sequence number>, can then be checked for correctness.

An Example of Arbitrating a Signature

Let T denote the text of the ith message communicated from A to B, where M is given by

$$M = [A, B, i, T]$$

[13] Greater security can be achieved if AC is defined as the concatenation of $Yn + 1$ and $Y'n + 1$, where $Yn + 1$ and $Y'n + 1$ are the last blocks of ciphertext that result when X0, X1, ..., Xn is enciphered with two different keys, K and K′, respectively. This would thwart the described attack in which the opponent generates random perturbations on only two starting messages (see Compressed Encoding Function, and also reference 10).

At A, data (RN, M, ¤) is enciphered using cryptographic key KA and initializing vector Z = 0 to produce ciphertext (Y0, Y, AC), where RN (defined X0 above) is a 64-bit random number, M = X1, X2, . . . , Xn, and ¤ = RN + X1 + X2 + . . . + Xn (mod 2^{64}). Data (A, Y0, Y, AC)—A's unencrypted identifier and the just produced ciphertext—is then sent to the arbiter. Identifier A is included in the transmission so that the arbiter can identify and obtain KA from his key table. (Recall that <sender ID> in M is encrypted.)

At the arbiter, ciphertext (Y0, Y, AC) is deciphered using KA. To prevent messages from being sent under an assumed identity, the arbiter checks that <sender ID> matches the received, unencrypted ID used to retrieve KA (i.e., that <sender ID> = A). If the two identifiers are unequal, an appropriate response is communicated to the sender and the procedure halted. If the two identifiers are equal, the arbiter checks that the last block of recovered plaintext equals RN + X1 + X2 + . . . + Xn (mod 2^{64}), and if so, it is concluded:

1. (Y0, Y, AC) was enciphered using KA and Z = 0, and therefore message M originated with A.
2. M is the true content of the communicated message.

If the last block of recovered plaintext is unequal to RN + X1 + X2 + . . . + Xn (mod 2^{64}), the message is not considered genuine. The arbiter sends the following message to B:

$$M' = [A, B, i, T']$$

where

$$T' = [T, RN, AC, \langle\text{genuine}\rangle] \text{ if M is genuine,}$$

or

$$T' = [T, RN, AC, \langle\text{not genuine}\rangle] \text{ if M is not genuine.}$$

In the example, text T, RN, and AC are repeated in their entirety, whereas <genuine> and <not genuine> denote established code words. A new random number, RN′, is selected and data (RN′, M′, ¤′) are enciphered with key KB and initializing vector Z = 0 to produce ciphertext (Y0′, Y′, AC′). The ID of the receiver, which is included in the received message, is used to locate KB in the arbiter's key table. Data (Y0′, Y′, AC′) are then sent to B.

Having established a communication session with the arbiter, B deciphers (Y0′, Y′, AC′) using Z′ = 0 and KB, and then authenticates M′ in the same way that M was authenticated by the arbiter. B accepts M if and only if

1. M′ is genuine.
2. Code word <genuine> is received in T′.
3. M′ satisfies all other conditions imposed by the digital signature procedure (e.g., parameters <sender ID>, <receiver ID>, and <sequence number> are correct).

RN, M, and AC are retained by B in order to prove later that M was received from A. This can be done by requesting the arbiter to encipher (RN, M, ¤) using Z = 0 and KA (obtained from the arbiter's key table), and then verifying that the last block of produced ciphertext is equal to the value AC (held by B).

In the example, AC is a function of both the message and the sender's secret key. It can be computed by either the arbiter or the sender, but not by the receiver. In that case, AC is the *digital signature* of the message.

A Weak Approach

Consider an approach in which X0 is eliminated and Z is produced via a generator of pseudo-random numbers. The process of encipherment is as follows:

$$Y1 = E_K(X1 \oplus Z)$$
$$Y2 = E_K(X2 \oplus Y1)$$
$$\vdots$$
$$Yn + 1 = E_K(Xn + 1 \oplus Yn)$$

where $Xn + 1 = X1 + X2 + \ldots + Xn \pmod{2^{64}}$.

At A, data (M,¤), where $M = X1, X2, \ldots, Xn$ and $¤ = X1 + X2 + \ldots + Xn \pmod{2^{64}}$, are enciphered using cryptographic key KA and initializing vector Z to produce (Y, AC). Data (A, Z, Y, AC) are then sent to the arbiter. In the modified procedure, Z is sent to the arbiter instead of Y0.

The procedure at the arbiter is unchanged, except that Z instead of RN is sent in T′:

$$T' = [T, Z, AC, \langle\text{genuine}\rangle] \text{ if M is genuine,}$$

or

$$T' = [T, Z, AC, \langle\text{not genuine}\rangle] \text{ if M is not genuine.}$$

Data (M′, ¤′) are enciphered with key KB and a new initializing vector Z′, which produces ciphertext (Y′, AC′). Data (Z′, Y′, AC′) are then sent to B. The procedure at B is unchanged, except that Z, M, and AC are retained in order to prove later that M was received from A.

This approach allows users to forge messages and signatures.[14] For example, user A could create a forged message and signature, purportedly from user B, as follows. User A sends (A, Z, Y, AC) to the arbiter, where

[14] This weakness was pointed out by Miles Smid, National Bureau of Standards.

(Y, AC) = ciphertext produced from plaintext (M, ¤) using KA

M = [A, B, i, T]

¤ = the 64-bit quantity produced by adding the blocks in M modulo 2^{64}

Except in this case T contains the following:

T = [pad bits, Mf, ¤f]

where

pad bits = enough bits to align Mf on a block boundary

Mf = [B, A, j, Tf] is the forged message

j = sequence number of some message received previously from user B

¤f = the 64-bit quantity produced by adding the blocks in Mf modulo 2^{64}

Thus,

T = [pad bits, B, A, j, Tf, ¤f]

After validating (Y, AC), the arbiter sends (Z′, Y′, AC′) to user B, where

(Y′, AC′) = the ciphertext produced from plaintext (M′, ¤′)

M′ = [A, B, i, T′]

T′ = [T, Z, AC, <genuine>]

Or, upon substituting for T, M′ equals

M′ = [A, B, i, pad bits, B, A, j, Tf, ¤f, Z, AC, <genuine>]

However, user A intercepts (Z′, Y′, AC′) (e.g., by using a wiretap) and jams the transmission to prevent it from reaching user B. The intercepted ciphertext is then used by user A to produce (Zf, Yf, ACf), where

Zf = X(i) ⊕ Y(i); Y(i) is the block of ciphertext produced from X(i) and X(i) is the block of plaintext just prior to the second B in M′

Yf = that portion of the intercepted ciphertext corresponding to Mf

ACf = that portion of the intercepted ciphertext corresponding to ¤f

(Zf, Mf, ACf) can now be given to the arbiter as proof that Mf was received from user B.

The weakness is overcome when Z is held constant and a random X0 is appended to the front of the message as described above.

Additional Weaknesses

There are encryption methods that permit message authentication which would have undesirable properties if used in a procedure for arbitrated signatures. For example, if a form of chained block encryption with plaintext-ciphertext feedback (see Figure 2-16) was used, it would be possible under certain conditions for the sender and receiver to forge messages and signatures.

Consider an implementation in which a message, M = X1, X2, . . . , Xn, is enciphered as:

$$Y1 = E_K(X1 \oplus Z)$$
$$Y2 = E_K(X2 \oplus X1 \oplus Y1)$$
$$\vdots$$
$$Yn + 1 = E_K(Xn + 1 \oplus Xn \oplus Yn)$$

where K is the cryptographic key, Xn + 1 = Z (Z is a random initializing vector), and Yn + 1 is defined as the AC. This encryption method has the peculiar property that for any value D the encipherment of M′ = X1 ⊕ D, X2 ⊕ D, . . . , Xn ⊕ D using cryptographic key K and initializing vector Z′ = Z ⊕ D is the same as the encipherment of M = X1, X2, . . . , Xn using cryptographic key K and initializing vector Z (i.e., Y′1 = Y1, Y′2 = Y2, . . . , Y′n + 1 = Yn + 1). Consequently, there are numerous easily derived messages that will produce the same AC.

The sender might be able to find values X1, X2, . . . , Xn and D (especially if n is small) such that both M and M′ are valid messages, except that the receiver would be willing to accept M whereas he would be unwilling to accept M′. The sender would send M to the receiver and later claim that M′ was sent.

On the other hand, if a protocol is used in which Xn + 1 is equal to a constant, say all 0 bits, the arbiter could be tricked into accepting an altered message. Let (Z, M, AC) denote a valid triple (initializing vector, message, authentication code) received by the receiver from the arbiter, and let M = X1, X2, . . . , Xn. The receiver can produce a forged triple (Zf, Mf, ACf) provided that he knows the value of some X′1 and Y′1 such that Y′1 = E_K(X′1). X′1 and Y′1 may be determined from previous transmissions. In that case, Z is replaced by X′1 ⊕ X1 ⊕ Y′1 ⊕ Y1 and X1 is replaced by X1 ⊕ Y′1 ⊕ Y1:

$$Zf = X'1 \oplus X1 \oplus Y'1 \oplus Y1$$
$$Mf = X1 \oplus Y'1 \oplus Y1, X2, X3, \ldots, Xn$$

and ACf = AC. Observe that the feedback used to encipher block Xi (i =

2, 3, . . . , n + 1) is Xi − 1 ⊕ Yi − 1 when (X1, X2, . . . , Xn + 1) is enciphered with Z and K, and when (X1 ⊕ Y'1 ⊕ Y1, X2, . . . , Xn + 1) is enciphered with Zf = X'1 ⊕ X1 ⊕ Y'1 ⊕ Y1 and K. Thus in either case the second, third, and so on, blocks of ciphertext are the same, and therefore, the same authentication code is produced—proving that (Zf, Mf, ACf) would be accepted by the arbiter.

USING DES TO OBTAIN PUBLIC-KEY PROPERTIES

Even though there are significant differences between public-key and conventional algorithms, a conventional algorithm can provide some of the properties of a public-key algorithm. Thus while the algorithms are markedly different, the cryptographic systems or functions provided the respective systems (i.e., what the user sees or perceives) may be very much alike.

The properties of public and private keys usually associated only with a public-key algorithm can be attained with a conventional algorithm provided that: (1) the public and private (DES) keys are used only at devices and equipment belonging to a specific cryptographic system, and (2) the secrecy and integrity of other system-managed keys stored within these cryptographic devices can be maintained.

Assuming a conventional algorithm, the basic problem involves the design of a trapdoor one-way function that can be used to encrypt/decrypt (transform) time-variant data (e.g., messages transmitted in a communication network). To some degree, the unidirectional cryptographic operations reported in reference 15, and further elaborated on in reference 16, were designed along these lines. Used for key management, one cryptographic operation encrypts keys sent to other devices and a second operation decrypts keys received from other devices. In addition, the two cryptographic authentication operations reported in reference 17, which allow time-variant quantities to be authenticated via a special precomputed test pattern, were also designed around this principle.

A method for implementing unidirectional DES data-encrypting keys was first proposed by Smid [18]. A data key Kij is used to transmit data from user i to user j (i.e., only user i can encrypt with Kij and only user j can decrypt with Kij). A second data key, Kji, is used to transmit data from user j to user i. With this scheme, the key assumes either a transmit (encipher only) or a receive (decipher only) capability. This method permits a protocol for digital signatures to be implemented, similar to that obtained with a public-key algorithm.

A Key Notarization System for Computer Networks[15]

System Design

The key notarization system is designed for computer networks which consist of host computers, user terminals, and key notarization facilities. The

[15] Reprinted in part from NBS Special Publication 500-54, October 1979 [18].

key notarization facility is analogous to the cryptographic facility described in Chapter 4 or to the security module in reference 19.

The host controls the normal operation and communication of the terminals. Terminals have the capability of communicating with the host, with other local terminals through the host, and with terminals of other hosts through communication channels called *interchanges*. Each terminal can use the host key notarization facility by means of user commands. All commands are implemented in the key notarization facility, and every key notarization facility has the capacity to generate keys for distribution to other hosts or facility users.

Two distinct types of keys are used: *interchange keys* (**IK**) and *data keys* (**DK**). Interchange keys and data keys are similar to the secondary and primary keys introduced in Chapter 4. Interchange keys are used for the exchange of data keys between users. One interchange key, called the *facility interchange key*, is used for the encryption of facility user passwords. Other interchange keys may be available for the exchange of data keys between facilities or for subgroups of a facility. Interchange keys are generated outside the network and are entered, unencrypted, directly into the key notarization facility. This permits two facilities to enter the same interchange key. Data keys are used to encrypt data and are generated in the key notarization facility.

The key notarization facility contains the DES algorithm, which employs a secret key to encipher/decipher data or other keys. It has a control microprocessor and memory to implement commands and data transfers. The key notarization facility must also store the unencrypted interchange keys and the states of active users. An *active state* consists of a user identifier, two initializing vectors, and two unencrypted data keys for transmitting and receiving data, respectively. A user is *active* as soon as his identifier is loaded into the key notarization facility. He may then proceed to load the rest of his state.

The key notarization facility contains a key generator which is capable of generating unpredictable keys. Once the 56-bit keys are generated, the proper parity is determined and the entire 64-bit key is encrypted before it is returned to the host. Thus no clear keys are known outside the key notarization facility. The key generator is also used to generate 64-bit initializing vectors which initialize the DES.

Identifiers and Key Notarization

Identifiers are nonsecret binary vectors of up to 28 bits that uniquely identify each user in the network. When a user first attempts to use the key notarization facility, he must submit his identifier along with the correct password to establish an active state in the key notarization facility. Both the host and the key notarization facility employ identifiers to recognize users.

Let i and j be identifiers, and let K be a 64-bit DES key. (i || j) represents the concatenation of i and j. K consists of 8 bytes, each byte containing 7 information bits and a parity bit. K XOR (i || j) is a special function defined as follows. The leftmost 7 information bits of K are Exclusive-ORed with the

leftmost 7 bits of i. The eighth bit, a parity bit, is then appended so the modulo two sum of the eight bits is odd. Then the next 7 information bits of K are Exclusive-ORed with the next 7 bits of i and the correct parity bit is appended. This continues until the last 7 information bits of K have been Exclusive-ORed with the last 7 bits of j and the final parity bit has been set. Therefore, K XOR (i || j) is a valid DES key with 56 information bits and 8 parity bits. All passwords and data keys are encrypted under K XOR (i || j) for some K and some i, j pair. In the case of passwords, i = j.

When key notarization is used, keys and passwords are sealed upon encryption by the key notarization facility with the identifiers of the transmitter (or key generator) and the receiver. To generate a notarized key, the transmitter must identify itself to the key notarization facility and provide proof of its identity by supplying the correct password. This is called user authentication. The transmitter must also specify the intended receiver of the key.

A generated key is immediately encrypted under an interchange key Exclusive-ORed with the proper identifier pair, that is IK XOR (i || j). The identifier of the user requesting the key, who is also the transmitter, is always the left identifier (i), and the identifier of the intended receiver is the right identifier (j) in the identifier pair.

Once encrypted, the correct key cannot be decrypted unless the correct identifier pair is again provided. To decrypt the key, the receiver identifies itself and provides password proof of its identity. The receiver must also supply the identifier of the transmitter. If the identification information is not the same as that provided by the transmitter to its key notarization facility, then the decrypted key will not equal the original key and no information can be correctly decrypted. Thus the receiver must know the correct transmitter and be the intended receiver.

User Authentication

Each user has a password used to authenticate and permit him to invoke user commands. The plain password is passed through an encryption function, involving the users identifier, and the result is compared in the key notarization facility with a stored value before the user is activated. No other command will be accepted by the key notarization facility until the user's identity has been authenticated. Each user's password is stored in system memory encrypted under the facility interchange key combined with the users identifier. It is assumed that the host can maintain the correct identity of a user once he has been authenticated. Thus after users are activated, they need not resubmit their passwords each time a new command is issued.

Let $E_K(X)$ indicate the encryption of X under key K using the electronic codebook mode of DES operation [20]. Thus $E_{(IK1 \text{ XOR } (i||i))}(PWi)$ denotes the encryption of PWi under IK1 Exclusive-ORed with user i's identifier pair, (i || i). IK1 is used because the passwords are from the system memory of host 1 and IK1 is the facility interchange key of host 1.

To protect against substitution, the password is encrypted under IK1 Exclusive-ORed with the appropriate identifier pair. If identifiers were not

used and user j could gain access to the system memory, he might substitute his own encrypted password, $E_{IK1}(PWj)$, for user i's encrypted password $E_{IK1}(PWi)$. User j could then be authenticated as user i by submitting his own password and claiming to be user i. However, if an identifier pair is Exclusive-ORed with the interchange key in the manner described, then $E_{(IK1 \text{ XOR } (i \| i))}(PWj)$ would be calculated upon authentication and it would not compare with $E_{(IK1 \text{ XOR } (j \| j))}(PWj)$ which was submitted as user i's encrypted password.

User i's memory contains personal and shared data keys. *Personal data keys* are encrypted under the facility interchange key Exclusive-ORed with the user's identifier pair. Personal keys may be used to encrypt files and other private data, but they cannot be shared. User i's memory also contains shared data keys. A *shared data key* is encrypted under an interchange key Exclusive-ORed with the concatenation of user i's identifier and another user's identifier, say user j. (i || j) uniquely identifies the communication parties. If (i || j) were not used, another user could substitute his own data key encrypted under the interchange key and could then decrypt any subsequent ciphertext encrypted with that data key. Similarly, when user j receives $E_{(IKp \text{ XOR } (i \| j))}(DKij)$, he must know that he is communicating with user i over interchange p to decipher DKij correctly. Thus the transmitter is prevented from posing as someone else. And since several users may all use the same IKp to communicate, this protection is critical.

Commands

The key notarization facility supports the required commands through a combination of hardware and software. The commands are used for:

1. Data encryption, decryption, and authentication.
2. Password initialization, notarization, change, and reencryption.
3. Reserving and logging out of active user states.
4. Generating notarized data keys and initializing vectors.
5. Loading notarized data keys and initializing vectors.
6. Reencrypting data keys.

A user must authenticate and receive an active state before he can execute any other commands. Password initialization is the only command function which must be privileged to the security officer.

Digital Signatures

Since the key notarization facility combines identifiers with interchange keys for protection against substitution, and employs a separate encryption and decryption key storage, one cannot encrypt data in a key that was generated by another user. Therefore, any encrypted message may be regarded as a signed message (or a signature). It is assumed that messages can be distinguished (e.g., by including redundant data in them). No additional keys or commands are required. All user j needs to do is keep $E_{(IKp \text{ XOR } (i \| j))}(DKij)$, $E_{DKij}(IV)$,

and the encrypted message in order to be able to prove that it was received from user i. Of course, user j may send a signed message to user i using data key DKji (≠ DKij).

Suppose user i generates a data key for communication with user j. The encrypted key would be of the form

$$E_{(IKp \text{ XOR } (i \,||\, j))}(DKij)$$

where IKp is the interchange key for interchange p and DKij is the data key generated by user i for transmission to user j. Whenever user i generates a key, his identifier is always leftmost in the identifier pair. The only way user j can load DKij is by loading it as a receive key. If user j tries to load DKij as a transmission key (i.e., for the encryption of data going to user i), the key notarization facility will use (j || i) instead of (i || j) when decrypting DKij. If user j tries to load the key as a personal key, then (j || j) will be used. When DKij is loaded as a receive key, only the decryption commands have access to it. Since there is no way for user j to get DKij into the transmit data key active storage, he cannot encrypt a message under DKij or alter a message already encrypted under DKij.

If user j generates a data key for communication with user i, the key will be of the form

$$E_{(IKp \text{ XOR } (j \,||\, i))}(DKji)$$

However, user j cannot claim that a message encrypted under DKji came from user i, since he could be challenged to decrypt the encrypted message. To do so, user j would have to load DKji by submitting $E_{(IKp \text{ XOR } (j \,||\, i))}(DKji)$ to the key notarization facility and claim to be the receiver. The key notarization facility would not load the correct data key (DKji), since (i || j) instead of (j || i) would be used as the identifier pair when decrypting $E_{(IKp \text{ XOR } (j \,||\, i))}(DKji)$. Thus the decrypted message would be garbled.

It is assumed that the key notarization facility of each host is physically secure from all users and that shared interchange keys are securely distributed. One must guard against disclosure and substitution of keys. If one could gain knowledge of the shared key, one could forge all signatures sent between both facilities. Of course, all data keys encrypted under the shared key would also be compromised.

A Method Using Variants of the Host Master Key[16]

A different method for implementing encipher only and decipher only keys with the DES is discussed below. KE (the encipher only key) and KD (the decipher only key) are defined as follows:

$$KE = E_{KMI}(K)$$

$$KD = E_{KMJ}(K)$$

[16]©1980 Horizon House. Reprinted in part from INTELCOM '80 Conference Proceedings [21] with permission of Horizon House Telecommunications, Inc.

where KMI and KMJ are two different variants of the host master key (KM0), unique to the specification of KE and KD respectively, and K is a DES data-encrypting key.

Two new cryptographic operations are also defined to the cryptographic facility: *encipher data only* (**ENCO**) and *decipher data only* (**DECO**), as discussed below:[17]

$$\textbf{ENCO: } \{\textbf{KE, X}\} \longrightarrow E_{D_{KMI}(KE)}(X)$$

$$\textbf{DECO: } \{\textbf{KD, Y}\} \longrightarrow D_{D_{KMJ}(KD)}(Y)$$

where X and Y are the plaintext and ciphertext, respectively. ENCO uses variant KMI and DECO uses variant KMJ.

X is recovered whenever it is enciphered with KE and the result is deciphered with KD:

$$\text{DECO: } \{\text{KD, ENCO: } \{\text{KE, X}\}\} = \text{X}; \quad \text{for all X,}$$

and X is recovered whenever it is deciphered with KD and the result is enciphered with KE:

$$\text{ENCO: } \{\text{KE, DECO: } \{\text{KD, X}\}\} = \text{X}; \quad \text{for all X}$$

provided that

$$D_{KMI}(KE) = D_{KMJ}(KD),$$

which is the case whenever $KE = E_{KMI}(K)$ and $KD = E_{KMJ}(K)$.

For an arbitrary value of K, $E_{KMI}(K)$ is generally unequal to $E_{KMJ}(K)$ (i.e., $KE \neq KD$). This means that when plaintext is enciphered/deciphered with a given key parameter (KE or KD), it cannot be recovered (deciphered/enciphered) with the same key parameter.

Because of the complexity of DES, a knowledge of KE does not reveal KD (and vice versa), even though KMI and KMJ are variants of the same key (KM0). But when used in conjunction with the cryptographic operations ENCO and DECO, KE permits data to be enciphered under a secret DES key (K) and the corresponding KD permits data to be deciphered under the same secret DES key (K). Therefore, as far as the cryptographic system is concerned, enciphering and deciphering are performed with K. But as far as the cryptographic system users are concerned, enciphering is performed with KE and deciphering is performed with KD. That is, the keys and the attributes of those keys are defined respectively different to the user and the cryptographic system.

The properties of a public-key cryptosystem are achieved by making KE (the encipher only key) public and keeping KD (the decipher only key) pri-

[17] ENCO and DECO illustrate the Electronic Codebook (ECB) mode of DES operation [20]. The encipher only and decipher only attributes could also be extended to cover encryption and decryption using block chaining techniques, although such methods are omitted from this discussion.

vate. The approach, however, is not as flexible as a true public-key cryptosystem. KE and KD can be used only at the host system where they were originally created. Moreover, the encipher only and decipher only properties of the keys are guaranteed only as long as the secrecy of KM0 or any of its variants, and the data-encrypting key associated with each KE and KD pair, can be guaranteed.

To allow a user to create a public and private key pair, a key generation function is defined called *generate key* (**GKEY**):

$$\textbf{GKEY}: \{\ \} \longrightarrow E_{KMI}(K), E_{KMJ}(K)$$

This operation has no input parameters, and when invoked a random number K is generated within the cryptographic facility and in turn enciphered first under KMI and second under KMJ. The variants KMI and KMJ are derived inside the cryptographic facility by a selected inversion of bits of the host master key KM0.

The principle of encipher only and decipher only keys can also be extended to a network of interconnected terminals and host computers [22]. However, the details are omitted from this discussion.

LEGALIZING DIGITAL SIGNATURES[18]

In the absence of an omnibus statute governing paperless commercial transactions via an electronic communications network, parties are free to enter into their own agreements. However, if disagreements later arise, the party seeking to enforce the contract will prevail only if the agreement complied with certain basic legal requirements. These requirements may include the the provisions of statutes of frauds (as imposed by the UCC and/or local law), acknowledgments, recording, and reasonableness. Modern statutes of frauds require some writing which indicates that a contract for sale has been made between the parties at a defined price, that it reasonably defines the subject matter, and that it is signed by either the party against whom enforcement is sought or by his duly authorized agent. We have seen, however, that the signed requirement may be satisfied by something less than a formal handwritten signature. A mere pattern of bits, whether in clear or encrypted form, would not as a practical matter serve as the required symbol in lieu of a handwritten signature, even though the pattern of bits was transmitted or accepted by a party with the intention at that time of authenticating a writing. This is because a pattern of bits which is used as a signature may be altogether too easily manipulated or forged, and is not part of, or annexed to, a tangible writing. Moreover, unless the pattern of bits were predefined to have a particular meaning to the party receiving it, or unless an established code form were adopted by the parties, it would be utterly without meaning.

Therefore, it seems doubtful that, by itself, a special pattern of bits trans-

[18]©1978 McGraw-Hill, Inc. Reprinted in part from *Data Communications,* February 1978 [1].

mitted together with a message and subsequently recorded on some machine-readable medium would satisfy the necessary legal requirements of signature. On the other hand, when an initial written agreement, signed in the ordinary sense, is entered into by the parties in question, it appears that the legal requirements of signature can be satisfied. The initial written agreement in this case defines the procedures and protocols whereby the parties would conduct a series of future transactions, together with an agreed means and procedure for recording the elements of such transactions.

The UCC specifically authorizes parties to vary the provisions of the code by agreement, except as otherwise stated, and provided that the obligations of good faith, diligence, reasonableness, and care as prescribed by the code may not be disclaimed [Sec. 1–102–(3)]. The UCC further provides for parties involved in banking transactions to stipulate or agree to deviate from its requirements, and to determine for themselves the standards by which their responsibilities are to be measured, provided that a bank may not disclaim responsibility for its own lack of good faith or failure to exercise ordinary care or limit the measure of damages for such lack or failure [Sec. 4–103]. It is under this exception that banks have been able to operate current electronic funds transfer systems, including transactions with their customers and with other banks.

Of course, there are certain classes of transactions for which only accepted paper-based conventions will suffice. For example, to be enforceable, transactions involving real property must (in most states) be in writing, and be acknowledged and recorded in a public registry (the office of the county clerk in which the property is located). Hence to comply with present law, contracts of this nature could not be handled by electronic communications networks with a capability for digital signatures.

Initial Written Agreement

With each of the methods for obtaining digital signatures previously described, each party possesses certain secret and possibly nonsecret, information used in generating his own signature, and other nonsecret information used in checking or validating the signatures of others.

For this protocol to be workable, there must be some mechanism for each party to authenticate independently the nonsecret signature validation information which he holds. This could be done if each party were to record his own signature validation information at some established registry with recognized and accepted integrity, such as an office of the county clerk or the office of a secretary of state. Alternatively, one could include this information within the initial written agreement itself, which was shown to be necessary in order to comply with the underlying legal requirements for conducting signed transactions via an electronic network. Recall that in a public-key cryptographic system, the private deciphering key cannot be efficiently derived from the public enciphering key. Likewise, in a DES-based protocol, the private signature keys cannot be efficiently derived from the corresponding public validation quantities.

As to the question of whether a person who transmits a message signed

with an electronic digital signature is in fact authorized, the procedures necessarily imply that only an authorized agent would have access to the secret information needed to generate the signature. Thus, when the secret information used in generating signatures is stored within a computing system, the burden is upon installation management to assure that this information is kept secret, and that an adequate access control mechanism is in place so that signatures can be created only by authorized users (persons, programs, and the like).

Whoever has access to a principal's secret signature generation information will be deemed to be the principal's authorized agent. Therefore, installation management must also implement sufficient security controls in order to be alerted if this secret signature generation information should become exposed, or if the capability to sign messages has been obtained by unauthorized users. Failure of one of the principals to notify other parties that his digital signatures have been compromised may be deemed his own negligence, and might defeat any defenses he may later raise as to the authority of his agents.

Choice of Law

As part of their initial written agreement, the parties must specify a particular jurisdiction under whose laws the agreement is to be governed (such as New York law), and the forum for the litigation of disputes that may arise out of transactions executed via the electronic communications system. Where the parties agree to communicate via a common network and the information needed to validate signatures has been recorded or registered, the jurisdiction wherein such registry is located would be the reasonable and logical choice. Both interstate and international transactions may be accommodated in this manner.

The statute of limitations defines the period of time within which a lawsuit must be commenced from the time a cause of action accrues. In disputes involving contracts, the period in most states is six years. A cause of action upon a contract may accrue at the time the original written agreement was entered into, or at some time thereafter, when a signed message is transmitted. It would appear necessary, therefore, that both parties to a transaction (sender and receiver) retain all data relating to their initial written agreement and to each subsequent signed message for at least the period of the applicable statute of limitations.

Moreover, it would serve the interest of both parties to have a trusted mechanism for the recording of the time and date of the transaction. If the time and date were included as part of the message's content, the receiver would have a means of verifying and proving the time and date of the transaction. This, of course, could be easily accomplished with existing message time-stamping facilities already available in data processing systems.

Regardless of the protocol for implementing electronic digital signatures, the claim is made that if the protocol is implemented as intended, then one can be assured that (1) the sender is not able to later disavow messages as his own, (2) the receiver is not able to forge messages or signatures, and (3) both the sender and receiver are certain that the identity of the sender and the

contents of the message can be proved before a referee. As a consequence, the following may be said about the judicial acceptance of the electronic digital signature.

The parties may agree or stipulate as part of their initial written agreement that they will be bound by their digital signatures, that they agree to submit all disputes to a referee, and that they agree the concept of digital signatures is cryptographically sound. However, this agreement will not prevent one of the parties from later raising the claim that the indicated result lacks validity, that he did not understand the underlying scientific principle (not an unreasonable assertion), or that he was forced to sign the stipulation as a condition of his transacting business with the other party. As a practical matter, therefore, it is prudent to assume that such disputes will inevitably arise.

While various techniques exist for proving the validity of written signatures (ranging from expert handwriting analysis to the unique properties of handwritten signature acceleration patterns), the resolution of disputes over digital signatures will be based on validation quantities, the cryptographic strength of the algorithms used in the generation of signatures, and the like. Thus as part of the process of judicial acceptance, the courts must initially pass upon the question of the soundness of the underlying cryptographic technique.

All scientific aids and devices go through experimental and testing phases. During these phases there may be considerable scientific controversy over the validity of the technique, aid, or device. During this period of controversy, there is the danger that a trial of a legal dispute between the parties may result in the trial of the validity of the new scientific technique, rather than a trial of the issues involved in the case. "It is not for the law to experiment but of science to do so" [*State vs. Cary*, 99 N.J. Sup. 323, 239 A.2d 680, aff'd, 56 N.J. 16, 264 A.2nd 209 (1970)].

When scientific aids to the discovery of truth receive general recognition within the relevant scientific community as to their accuracy, courts will not hesitate to take judicial notice of such fact and admit evidence obtained through their use. Judicial notice means that the underlying scientific principle upon which the new device or process is based need not be proved each time the results of the device or process are introduced into evidence.

Judicial Notice Recognized

As an example of judicial notice, each time a police officer testifies that according to the output display of his radar device the defendant was speeding, he need not present expert witnesses to testify to the scientific foundation of radar—that the radar transmitter and receiver can measure the velocity of a moving target based upon the Doppler effect of reflected waves. Radar has now become generally accepted as a means of measuring vehicle speed. All the officer must prove is that, on the particular occasion in question, his particular radar unit was properly set up, calibrated, and operated.[19] Examples

[19] A highly publicized case in Dade County Florida in 1979 regarding the reliability of radar devices and their use by police officers resulted in a ruling "that the reliability of radar speed measuring devices as used in their present modes and particularly in these cases, has not been established beyond and to the exclusion of every reasonable doubt." [23] Subsequent court decisions, however, have generally upheld the accuracy and reliability of police traffic radar.

of other scientific principles that have been reduced to practice and are now judicially noticed include the unique properties of fingerprints and ballistics comparisons [24].

While the digital signature concept could be implemented using any encryption algorithm, its own scientific acceptance would be aided by basing the scheme upon a strong encryption algorithm which itself has already been scientifically recognized and accepted. Of course, if and when additional scientific evidence becomes available to challenge the effectiveness of the algorithm, courts may later reject their reliance upon once accepted principles.

It would be preferable, therefore, for the digital signature to be based on an algorithm whose strength has been certified by the NSA. In all likelihood, this would satisfy the criteria for judicial acceptance of the validity of the underlying scientific principle of digital signatures, and could aid in the eventual acceptance by the courts of the digital signature concept.

REFERENCES

1. Lipton, S. M. and Matyas, S. M., "Making the Digital Signature Legal—and Safeguarded," *Data Communications,* **7**, No. 2, 41–52 (February 1978).
2. Corley, R. N. and Robert, W. J., *Dillavou and Howard's Principles of Business Law,* 9th ed., Prentice-Hall, Englewood Cliffs, NJ, 1971.
3. *Uniform Commercial Code,* 1972 Official Text with Comments, American Law Institute and National Conference of Commission on Uniform State Laws.
4. Diffie, W. and Hellman, M., "New directions in Cryptography," *IEEE Transactions on Information Theory,* **IT-22**, 644–654 (November 1976).
5. Merkle, R. and Hellman, M., "Hiding Information and Receipts in Trapdoor Knapsacks," *IEEE Transactions on Information Theory,* **IT-24**, 525–530 (September 1978).
6. Kohnfelder, L. M., *Towards a Practical Public-Key Cryptosystem,* BS Thesis, Department of Electrical Engineering, Massachusetts Institute of Technology, Cambridge (May 1978).
7. Rivest, R. L., Shamir, A., and Adleman, L., "A Method for Obtaining Digital Signatures and Public-Key Cryptosystems," *Communications of the ACM,* **2**, No. 21, 120–126 (February 1978).
8. Shamir, A., "A Fast Signature Scheme," Massachusetts Institute of Technology, MIT/LCS/TM-107 (July 1978).
9. Branstad, D., Davida, G. I., Hellman, M. E., Tuchman, W. L., and Sugarman, R., "On Foiling Computer Crime," *IEEE Spectrum,* **16**, No. 7, 31–49 (July 1979).
10. Yuval, G., "How to Swindle Rabin," *Cryptologia,* **3**, No. 3, 187–189 (July 1979).
11. Rabin, M. O., "Digitized Signatures," in *Foundations of Secure Computation,* edited by R. A. DeMillo, D. P. Dobkin, A. K. Jones, and R. J. Lipton, Academic Press, New York, 1978, 155–168.
12. Matyas, S. M. and Meyer, C. H., "Electronic Signature for Data Encryption Standard," *IBM Technical Disclosure Bulletin,* **24**, No. 5, 2332–2334 (October 1981).
13. Matyas, S. M. and Meyer, C. H., "Electronic Signature for use with the Data Encryption Standard," *IBM Technical Disclosure Bulletin,* **24**, No. 5, 2335–2336 (October 1981).
14. Matyas, S. M., "Digital Signatures—An Overview," *Computer Networks,* **3**, 87–94 (1979).
15. 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 Systems Journal,* **17**, No. 2, 106–125 (1978).

16. Lennon, R. E. and Matyas, S. M., "Unidirectional Cryptographic Functions Using Master Key Variants," *1979 National Telecommunications Conference Record*, **3**, 43.4.1–43.4.5 (November 1979).
17. Lennon, R. E., Matyas, S. M., and Meyer, C. H., "Cryptographic Authentication of Time-Invariant Quantities," *IEEE Transactions on Communications*, **COM-29**, No. 6, 773–777 (June 1981).
18. Smid, M. E., *A Key Notarization System for Computer Networks*, NBS Special Publication 500–54, U.S. Department of Commerce, National Bureau of Standards, Washington, DC (October 1979).
19. Campbell, C. M., Jr., "A Microprocessor-Based Module to Provide Security in Electronic Funds Transfer Systems," *Proceedings COMPCON 79*, 148–153 (1979).
20. *DES Modes of Operation*, Federal Information Processing Standard (FIPS) Publication 81, National Bureau of Standards, U.S. Department of Commerce, Washington, DC (1981).
21. Lennon, R. E., Matyas, S. M., and Meyer, C. H., "Cryptographic Authentication Methods for Emerging Data Processing Applications," *Proceedings, INTELCOM '80*, 337–341 (November 1980).
22. Lennon, R. E., Matyas, S. M., and Meyer, C. H., "Public-Key Enciphering/Deciphering Transformations Using a Conventional Algorithm," *IBM Technical Disclosure Bulletin*, **25**, No. 3A, 1241–1249 (August 1982).
23. *Police Traffic Radar*, U.S. Department of Transportation Issue Paper, DOT HS–805 254 (February 1980).
24. Maguire, J. M. and Chadborne, J. H., *Evidence—Cases and Materials*, 6th ed., Foundation Press, Mineola, NY, 1973.

Other Publications of Interest

25. Merkle, R. C., *Secrecy, Authentication, and Public Key Systems*, Technical Report No. 1979–1, Department of Electrical Engineering, Stanford University, Palo Alto, CA (June 1978).
26. Shamir, A., *A Fast Signature Scheme*, MIT Laboratory for Computer Science, Report TM–107 (July 1978).