Operating Systems

R. Stockton Gaines
Editor

# Using Encryption for Authentication in Large Networks of Computers

Roger M. Needham and
Michael D. Schroeder
Xerox Palo Alto Research Center

**Use of encryption to achieve authenticated communication in computer networks is discussed. Example protocols are presented for the establishment of authenticated connections, for the management of authenticated mail, and for signature verification and document integrity guarantee. Both conventional and public-key encryption algorithms are considered as the basis for protocols.**

**Key Words and Phrases: encryption, security, authentication, networks, protocols, public-key cryptosystems, data encryption standard**

**CR Categories: 3.81, 4.31, 4.35**

## Introduction

In the context of secure computer communications, authentication means verifying the identity of the communicating principals to one another. A network in which a large number of computers communicate may have no central machine or system that contains authoritative descriptions of the connected computers, of the purposes for which they are used, or of the individuals who use them. We present protocols for decentralized authentication in such a network that are integrated with the allied subject of naming. There is minimal reliance on network-wide services; in particular there is no reliance on a single network clock or a single network name management authority.

Three functions are discussed:

(1) Establishment of authenticated interactive communication between two principals on different machines. By interactive communication we mean a series of messages in either direction, typically each in response to a previous one.

(2) Authenticated one-way communication, such as is found in mail systems, where it is impossible to require protocol exchanges between the sender and the recipient while sending an item, since there can be no guarantee that sender and recipient are simultaneously available.

(3) Signed communication, in which the origin of a communication and the integrity of the content can be authenticated to a third party.

Secure communication in physically vulnerable networks depends upon encryption of material passed between machines. We assume that it is feasible for each computer in the network to encrypt and decrypt material efficiently with arbitrary keys, and that these keys are not readily discoverable by exhaustive search or cryptanalysis. We consider both conventional encryption algorithms and public-key encryption algorithms as a basis for the protocols presented.

We assume that an intruder can interpose a computer in all communication paths, and thus can alter or copy parts of messages, replay messages, or emit false material. While this may seem an extreme view, it is the only safe one when designing authentication protocols.

We also assume that each principal has a secure environment in which to compute, such as is provided by a personal computer or would be by a secure shared operating system. Our viewpoint throughout is to provide authentication services to principals that choose to communicate securely. We have not considered the extra problems encountered when trying to force all communication to be performed in a secure fashion or when trying to prevent communication between particular principals in order to enforce restrictions on information flow.

Our protocols should be regarded as examples that expose the authentication issues in large networks rather than as fully engineered solutions to the overall security problems of a particular application. While providing an adequate solution to the authentication problems specified and meeting most common security objectives, our protocols would need elaboration to meet other security goals such as preventing traffic analysis, withholding all matching cleartext-ciphertext pairs from an eavesdrop-

Permission to copy without fee all or part of this material is granted provided that the copies are not made or distributed for direct commercial advantage, the ACM copyright notice and the title of the publication and its date appear, and notice is given that copying is by permission of the Association for Computing Machinery. To copy otherwise, or to republish, requires a fee and/or specific permission.

Authors' present addresses: R.M. Needham, University of Cambridge Computer Laboratory, Corn Exchange Street, Cambridge, England; M.D. Schroeder, Xerox Palo Alto Research Center, 3333 Coyote Hill Road, Palo Alto, California 94304.

© 1978 ACM 0001-0782/78/1200-0993 $00.75.

per, and ensuring instantaneous detection of tampering, and also to maximize efficiency in particular networks. It is possible to devise other protocols similar to those presented that also meet the stated objectives.

There is a modest amount of literature on our subject, and methods have been proposed for several of the individual functions we describe [1, 3, 5, 6], although no work is reported that integrates these techniques and applies them in a decentralized environment, or that provides functionally equivalent protocols based on both conventional and public-key encryption.

## 1. Encryption Algorithms

The important difference between conventional and public-key encryption algorithms is the way keys are used. With a conventional encryption algorithm, such as the NBS Data Encryption Standard [7], the same key is used for both encryption and decryption. Authentication depends upon the two participants in a conversation being the only two principals (apart possibly from trusted servers) who know the key that is being used to encrypt the transmitted material. With a public-key encryption algorithm, a concept originated by Diffie and Hellman [3], two keys are necessary: one that is used in the conversion of cleartext to ciphertext, and another that is used in the conversion of ciphertext to cleartext. Furthermore, knowledge of one key gives no help in finding the other, and the two keys will act as inverses for each other. Elegant systems may be devised in which each principal has one public key and one secret key. Anyone may encrypt a communication for $A$ using his public key, but only $A$ can decrypt the result using his secret key. Likewise, only $A$ can encrypt messages that will decrypt sensibly with $A$'s public key. The first example of a public-key encryption algorithm was devised by Rivest et al. [9], and others are sure to follow.

## 2. Authentication Servers

With both kinds of encryption the basis of authenticated communication is a secret key belonging to each principal using the network, and there is need for an authoritative source of information about these keys. We use the term *authentication server* for a server that can deliver identifying information computed from a requested principal's secret key.

Since the main database of an authentication server is indexed by name, the management of authentication servers is related to the management of names. In an extended network it is inexpedient to have a single central name registration authority, so we suppose that there are multiple naming authorities, each of which assigns and cancels names as it wishes. With this organization, principals have names of the form "NamingAuthority.SimpleName." Associated with each naming authority are one or more name lookup servers and one or more authentication servers.[1]

A name lookup server is prepared to provide various network addresses associated with a given SimpleName, for example, the address of that principal's mail system buffer. One or more instances of a master name lookup server will provide the network addresses of appropriate name lookup and authentication servers when given a naming authority's name. Authentication servers perform strikingly similar functions for the two classes of encryption algorithms; the differences will be brought out as they arise.

## 3. Means of Encryption

One significant issue in this area of study is where the encryption and decryption are done. Branstad [2] suggests that these actions take place in the network interface of a computer. It is a requirement of some of our protocols that the encryption be done elsewhere, because it is necessary to prepare an encrypted message without actually sending it yet or to receive an encrypted message without knowing at the network interface what the key is. Accordingly we have assumed that any hardware encryption aid is located so one can say

$X :=$ encrypt($Y$, $Key$)

and still have $X$ in hand, or say

**if** ($X :=$ decrypt($Y$, $Key1$)) = nonsense
**then** $X :=$ decrypt($Y$, $Key2$) **fi**

## 4. Protocols for Establishing Interactive Connections

### Protocol 1. With Conventional Algorithms

If a conventional algorithm is used then each principal has a secret key that is known only to itself and to its authentication server, the contents of which are accordingly secret. The essential step in setting up secure communication between $A$ and $B$ is for the initiator, say $A$, to generate a message with two properties:

(a) It must be comprehensible only to $B$, i.e. allow only $B$ to use its contents to identify himself to $A$.

(b) It must be evident to $B$ that it originated with $A$.

The use of encryption to achieve these properties was first described by Feistel [4] and applied to a network context by Branstad [1].

[1] Naming authorities are independent of network topology; they need have nothing to do with subnetworks or with particular computers on the network. Multiple identical name lookup servers and authentication servers for a single naming authority may be used to make sure that these services are topologically "close" to those needing to use them, and to enhance reliability. Our multiple authentication servers must be carefully distinguished from those proposed by Diffie and Hellman [3], which perform the quite different function of checking one another. In that case every user is registered with every authenticator, the aim being to defend against corruption of particular authenticators.

Assuming for the moment that $A$ and $B$ are in the purview of the same authentication server $AS$, we now outline a protocol. The notation used will be followed throughout: encryption is indicated by braces that are superscripted with the key used.

The protocol opens with $A$ communicating in clear to $AS$ his own claimed identity and the identity of the desired correspondent, $B$, together with $A$'s nonce identifier for this transaction, $I_{A1}$. ("Nonce" means "used only once.") Here the nonce identifier must be different than others used by $A$ in previous messages of the same type. The first message of the protocol is:

$$A \rightarrow AS: \quad A, B, I_{A1} \tag{1.1}$$

Upon receiving message (1.1), $AS$ looks up the secret, identifying keys of both parties and also computes a new key $CK$ that will be the key for the conversation if all goes well.[2] The next transaction is a rather complicated message from $AS$ *to* $A$:

$$AS \rightarrow A: \quad \{I_{A1}, B, CK, \{CK, A\}^{KB}\}^{KA} \tag{1.2}$$

where $KA$ and $KB$ are $A$'s and $B$'s secret, identifying keys. Because (1.2) is encrypted with $A$'s secret key, only $A$ can decrypt it and discover the conversation key $CK$. Following decryption, $A$ checks for the presence of the intended recipient's name, $B$, and the correct identifier, $I_{A1}$, in order to verify that the message really is a reply by $AS$ to the current enquiry. Both the name of the intended recipient and the transaction identifier must appear in message (1.2). If the recipient's name is left out, then an intruder could change that name in message (1.1), say to $X$, before $AS$ receives it, with the subsequent result that $A$ would unknowingly communicate with $X$ instead of $B$. If the identifier is left out, then an intruder could substitute a previously recorded message (1.2) (from $AS$ to $A$ about $B$) and force $A$ to reuse a previous conversation key.[3] $A$ remembers $CK$ and sends the part encrypted with $KB$ to $B$:

$$A \rightarrow B: \quad \{CK, A\}^{KB} \tag{1.3}$$

The real $B$, but no other, will be able to decrypt message (1.3) and emerge with the conversation key $CK$, the same as $A$ has. $B$ also knows the identity of the intending correspondent, as authenticated by $AS$.

It is worth reviewing at this point the state of knowledge of the two parties. $A$ now knows that any communication he receives encrypted with $CK$ must have originated with $B$, and also that any communication he emits with $CK$ encryption will be understood only by $B$. Both are known because the only messages containing $CK$ that have ever been sent are tied to $A$'s and $B$'s secret keys. $B$ is in a similar state, mutatis mutandis. It is important, however, to be sure that no part of the protocol exchange or ensuing conversation is being replayed by an intruder from a recording of a previous conversation between $A$ and $B$. In relationship to this question the positions of $A$ and $B$ differ. $A$ is aware that he has not used the key $CK$ before and therefore has no reason to fear that material encrypted with it is other than the legitimate responses from $B$. $B$'s position is not so good; unless he remembers indefinitely keys previously used by $A$ in order to check that $CK$ is new, he is unclear that the message (1.3) and the subsequent messages supposedly from $A$ are not being replayed. To guard against this possibility, $B$ generates a nonce identifier for the transaction, $I_B$, and sends it to $A$ under $CK$:

$$B \rightarrow A: \quad \{I_B\}^{CK} \tag{1.4}$$

expecting a related reply, say one less:

$$A \rightarrow B: \quad \{I_B - 1\}^{CK} \tag{1.5}$$

If this reply is satisfactorily received, then the mutual confidence is sufficient to enable substantive communication, encrypted with $CK$, to begin.

There are five messages in protocol 1. The number may be reduced to three by $A$'s keeping, for regular interaction partners, a cache of items of the form $B$: $CK$, $\{CK, A\}^{KB}$ derived from message (1.2), thus eliminating messages (1.1) and (1.2). Note however that, if such authenticators are cached, changes are needed to the protocol. With caching, the same $CK$ is being used again and again, so the conversation identifier handshakes need to be two-way, for example, by replacing steps (1.3) and (1.4) with:

$$A \rightarrow B: \quad \{CK, A\}^{KB}, \{I_{A2}\}^{CK} \tag{1.3'}$$

$$B \rightarrow A: \quad \{I_{A2} - 1, I_B\}^{CK} \tag{1.4'}$$

The change does not increase the number of protocol messages but does alter the content slightly. In practice, messages (1.3)–(1.5) would be used to start a two-way seriation in order to ensure the integrity of the subsequent conversation. Methods for ensuring integrity following initial contact have been studied by Kent [5].

### Protocol 2. With Public-Key Algorithms

We use key labels such as $PKA$ for $A$'s public key and $SKA$ for his secret one. The exchange opens with $A$ consulting the authentication server in the clear to find $B$'s public key.

$$A \rightarrow AS: \quad A, B \tag{2.1}$$

$AS$ responds with:

$$AS \rightarrow A: \quad \{PKB, B\}^{SKAS} \tag{2.2}$$

where $SKAS$ is the authentication server's secret key. $A$ is presumed to know the $AS$'s public key, $PKAS$, which is used to decrypt the message. $A$ must obtain and store $PKAS$ in a reliable way, so he is sure it is correct. If an

[2] The new key must be unpredictable and should never have been used before.

[3] Also note that messages (1.1) and (1.2) together, and others in our protocols, make available known plaintext encrypted with a principal's identifying key. If there is concern about cryptanalytic attack based on known plaintext being used to expose an identifying key, then an additional temporary key $TK$ may be used where appropriate throughout, so that $\{X\}^{KA}$ becomes $\{TK\}^{KA}\{X\}^{TK}$.

intruder somehow could provide an arbitrary value that $A$ thinks is $PKAS$, then that intruder could impersonate $AS$.

The importance of the reciprocity between the public and secret keys is shown here. Encryption of message (2.2) is required not to ensure the *privacy* of the information but to ensure its *integrity*. It is important that $A$ should be sure that he is getting $PKB$ rather than the public key of some miscreant. $A$ knows that the name of the intended recipient, $B$, was correctly communicated to $AS$ because that name is returned in message (2.2).

The next step is for the communication with $B$ to be initiated:

$$A \rightarrow B: \quad \{I_A, A\}^{PKB} \tag{2.3}$$

This message, which can only be understood by $B$, indicates that someone purporting to be $A$ wishes to establish communication, and secretly communicates a nonce identifier, $I_A$, generated by $A$. $B$ decrypts the message with his secret key and then finds $PKA$ with steps similar to (2.1) and (2.2):

$$B \rightarrow AS: \quad B, A \tag{2.4}$$

$$AS \rightarrow B: \quad \{PKA, A\}^{SKAS} \tag{2.5}$$

Message (2.5) is encrypted for integrity, as was (2.2), not for secrecy. At this point a double handshake is needed to authenticate $A$ and $B$ to one another and to establish the time integrity of the conversation. The handshake is completed as steps (2.6) and (2.7):

$$B \rightarrow A: \quad \{I_A, I_B\}^{PKA} \tag{2.6}$$

$$A \rightarrow B: \quad \{I_B\}^{PKB} \tag{2.7}$$

There are thus seven steps in this protocol as against five with protocol 1, but four of them (2.1, 2.2, 2.4, and 2.5) can be done away with by $A$ and $B$ both having local caches of commonly used public keys. The resulting three protocol steps have very similar purposes to the three remaining after caching in protocol 1.

Observe that, because public keys are not secret, double encryption, i.e. $\{\{message\}^{SKA}\}^{PKB}$, or some equivalent is required during the course of the ensuing interaction. If the data were simply encrypted with the public key of the recipient, then anyone else could inject material into the stream. An equivalent safeguard is to use an arbitrary number from a large space as the base for seriation of encryption blocks. This number may be initialized as $I_A$ or $I_B$ according to direction. An intruder would have no way of knowing what was the correct serial to insert in a forged packet, even if he had counted previous packets, since he could not know the correct base. The more bits that are devoted to this redundant seriation the fewer good data bits we get per unit decryption effort.

## 5. Multiple Authentication Servers

In the protocols just given we assumed that $A$ and $B$ were clients of the same authentication server. This restriction is not necessary, and we now remove it. When extending the protocols we must bear in mind that, while an authentication server must be regarded as the final authority for its clients, it must be able to have no effect for good or ill on communication between clients of other authentication servers. Then our system will not be upset completely by the conduct of a shoddy authenticator. Of course, outsiders will show circumspection on a human level in their dealings with a shoddy authenticator's clients.

The effects on the protocols of multiple authentication servers differ somewhat between the two encryption techniques. Consider first the case of conventional encryption. The requirement is still to produce an item of the form $\{CK, A\}^{KB}$ for $A$ to use when making his first approach to $B$ (see step (1.3)). To produce this quantity both authentication servers (which will be called $AS_A$ and $AS_B$) are involved, since only $AS_B$ can produce items encrypted with $KB$ and only $AS_A$ can produce items encrypted with $KA$. We find two more steps between (1.1) and (1.2), which constitute an interchange between the two servers. We suppose that separate measures have been taken to ensure secure communication between the servers—for example, their secret keys are held by a master server, and the regular servers establish secure links (by protocol 1 already given) whenever they come into operation. We also presume that names are, where necessary, always full "NamingAuthority.SimpleName" names, so that the correct authentication server can be located. As explained above, the knowledge of a naming authority's name leads to the network address of the associated authentication server.

$$AS_A \rightarrow AS_B: \quad CK, B, A, I_{A1} \tag{1.11}$$

$$AS_B \rightarrow AS_A: \quad \{CK, A\}^{KB}, I_{A1}, A \tag{1.12}$$

($I_{A1}$ is transmitted to avoid retention of state in $AS_A$ between messages (1.11) and (1.12).) Following (1.12) $AS_A$ is in a position to complete the protocol.

In the public-key case, since no secret keys are moved around, it is possible for $A$ to approach $AS_B$ directly if $A$ knows that server's public key. We assume that $A$ already has this knowledge, though in a strict case of total ignorance there would be key lookup steps, for example, correspondence with a master authentication server, before (2.1). With the knowledge of $PKAS_B$, $A$ corresponds directly with $AS_B$ in steps (2.1) and (2.2). Likewise, with knowledge of $PKAS_A$, $B$ corresponds directly with $AS_A$ in (2.4) and (2.5).

In both cases caching can be expected to reduce the number of protocol messages to three.

## 6. Implementing Authentication Servers

There are differences in the implementation of authentication servers for the two varieties of encryption. In the conventional case the content of the database, items of the form $A{:}KA$, must be kept secret (which could

be done by encrypting it with the secret, identifying key of the server). A secure transaction takes place every time the server is used: at step (1.2) the keys of both customers must be extracted in order to construct the message contents. By contrast, in the public-key case the content of the database need not be secret, and no secure transaction need take place when the server is used if the server's database is set up to contain items of the form $A$: $\{PKA, A\}^{SKAS}$ as required at step (2.2). (If the server contained the public keys directly, there would still be a secure operation at each use, for the reasons mentioned in the discussion of step (2.2).) With the public-key authentication server there still is a requirement for a secure computation, creating $\{PKA, A\}^{SKAS}$, but only when a new public key is registered, and this operation may be done outside the authentication server and the result added to the database in a nonsecure way. In practice, however, we suspect that the implementation of authentication servers would not differ as much as we have indicated, for reasons such as the need to prevent corruption of the public-key authentication server's data, which could prevent communication even though it will not lead to faulty authentication.

Note that with both encryption techniques the communications with servers can be done without the formalities of establishing what is usually called a "connection." The servers need never retain information about an ongoing transaction from one message to the next, so that repetition or loss of protocol packets does not matter. Only at step (1.11) does anything special have to be done to ensure lack of connection state. If this simplicity were lost, then the total cost of protocol exchanges would become higher.

## 7. One-Way Communication

In a computerized mail system it is impossible to depend upon interaction between the sender and the receiver in the course of each delivery. The mail is put into the hands of a transport mechanism and may be delivered later when the sender is no longer available. On the other hand, two-way authentication of sender and receiver is as desirable for mail as it is for interactive communication. Good design of a mail system would suggest that the mail transport mechanism not be part of the security system, and the proposals here meet that goal.

### With Conventional Algorithms

Consider a message used in a previous protocol:

$$A \rightarrow B: \quad \{CK, A\}^{KB} \tag{1.3}$$

This message has the property that if it be put at the head of mail encrypted with $CK$, then the whole is self-authenticating both as to recipient and originator even though $B$ played no part at all in the setting-up protocol. We assume that the subsequent individual blocks of the mail are securely seriated in, for example, the manner of Kent. The very fact of delay, however, causes special steps to be needed to ensure the time integrity of mail, i.e. that it has not been recorded by an intruder from an earlier transmission and repeated. We have avoided proposing the use of time-stamps elsewhere, because it presupposes a network-wide reliable source of time. Here there seems little alternative to the use of time-stamps; but it is possible to use them here without requiring a universal clock. A suitable technique is as follows. Each message has in its body a time-stamp indicating the time of sending. (Such a time-stamp is a normal part of most mail anyway.) The resolution needs to be fine enough that no two messages from the same source will have the same stamp. Any recipient, say $B$, maintains a register in which an entry of the form {*source*, *time-stamp*} is stored for each mail item received. A time interval $T$ is associated with $B$. $T$ is taken as an upper bound on clock asynchrony in the network and the interval between the time the mail was sent and the time of its arrival within $B$'s security control, after which time the mail cannot be diverted. A mail item is rejected if either its {*source*, *time-stamp*} is on the register or its time-stamp predates the current time by more than $T$. The register is kept small by discarding entries older than $T$. $T$ may vary dependent on $B$'s activity if a message may only arrive in his security control when he is present.

### With Public-Key Algorithms

The means of ensuring time integrity are identical in this case and will not be repeated. We have two alternative courses. With the first a header is sent that identifies $A$ to $B$ without using a handshake:

$$A \rightarrow B: \quad \{A, I, \{B\}^{SKA}\}^{PKB}$$

Here $A$ denotes the sender and $\{B\}^{SKA}$ enables authentication by $B$ of the identity of the sender using protocol transactions as at (2.4) and (2.5) (which may of course be short-cut by caching). $I$ is a nonce identifier that is used to connect the header with the ensuing message text sent under the protection of $PKB$, with a time-stamp as above and with a secure seriation as discussed earlier. The connection between header and message provided explicitly by $I$ in this protocol is provided implicitly by $CK$ in the case of a conventional encryption algorithm.

The other way to handle mail using the public-key system achieves the additional function of signature and is described in the next section.

## 8. Digital Signatures

The previous protocols are designed to authenticate each communicant to the other. It is sometimes necessary to provide evidence to a third party that a particular communication is exactly as received from a particular

sender. This requirement is met by signatures on paper documents. A common example is instructions from a superior to do something; the recipient needs to retain them as evidence that his actions were proper. To produce the analog of signed documents with messages, it is necessary that the recipient could not alter a signed text undetected and that the sender cannot credibly disclaim it. The ability to provide digital signatures depends upon there being something the originator can do which the recipient cannot.

**Protocol 3. Signatures with Conventional Encryption and a Little Help.**

One method uses a *characteristic function* of the cleartext message that is to be signed. The characteristic function must have the property that, given the cleartext message, the function, and the resulting characteristic value, it is hard to find another sensible cleartext message that produces the same characteristic value. It also is useful if the characteristic value is noticeably smaller than the cleartext message. Hard-to-invert transformations of the sort used to protect passwords [8] is a class of functions with the required properties.

While sending the text, say using the interactive or mail protocols described earlier, $A$ computes the characteristic value $CS$. He then requests a *signature block* from the authentication server:

$$A \rightarrow AS: \quad A, \{CS\}^{KA} \tag{3.1}$$

which the server supplies:

$$AS \rightarrow A: \quad \{A, CS\}^{KAS} \tag{3.2}$$

Message 3.2 is encrypted with $AS$'s key and therefore is accessible only to $AS$. Note that $A$ cannot validate the message, but if it has been interfered with, then $B$ subsequently will be unable to validate the signature, which he likely will do anyway before acting on the message if it contains instructions worthy of signature. $A$ sends the signature block to $B$ following the text to be signed.

On receipt $B$ first decrypts the text and computes its characteristic value, $CSC$. $B$ then communicates the signature block to the authentication server for decryption:

$$B \rightarrow AS: \quad B, \{A, CS\}^{KAS} \tag{3.3}$$

The server decrypts the signature block and returns its contents to $B$:

$$AS \rightarrow B: \quad \{A, CS\}^{KB} \tag{3.4}$$

If the returned $CS$ matches $CSC$, then the principal named in (3.4) is the sender of the signed text. $CSC$ not matching $CS$ could mean that any of the steps (3.1)–(3.3), or the association of the signature block with the signed text, has been interfered with. Earlier detection of certain types of interference is possible by using nonce identifiers in transactions (3.1)–(3.2) and (3.3)–(3.4). If $B$ wishes to retain the text as evidence, all he has to do is to retain the signature block and the text itself. In response to a challenge $B$ would produce the text and the signature block for an arbiter who would go through the communication of steps (3.3) and (3.4).

The extension of protocol 3 to the case of multiple authentication servers is straightforward.

**Signatures with Public-Key Encryption**

It is possible to provide signed text with a public-key system using a characteristic function as above. The public key system, however, provides another, more elegant, method that was first described by Diffie and Hellman. The first steps are for $A$ to find out $B$'s public key from cache or server, as before. The successive blocks of text, seriated for time integrity, are doubly encrypted:

$$A \rightarrow B: \quad \{\{text\text{-}block\}^{SKA}\}^{PKB}$$

$B$ can carry out the first decryption because of knowing $SKB$, and the second because of being able to find out $PKA$ by protocol exchange or from a cache. There is a need for header information to convey securely the identity of the originator so that $PKA$ can be correctly sought. $B$ is in no position to alter the content, since $SKA$ is not available to him. When challenged, $B$ simply performs the outer decryption on the whole text and passes the result to the arbiter who can use $PKA$ to finish the job. Note that the ability of an arbiter to perform his function seems to depend on $A$ not changing his key pair. Since such changes must be allowed as the only response to a key being compromised, it is necessary for the authentication server to retain a record of the old public keys of its principals and the time of the change, and for signed texts to contain the time that they were signed. An advantage of the signature protocol for conventional encryption algorithms is that an authentication server only need retain a record of changes to its own key to guarantee correct future arbitration.

## 9. Commentary

We conclude from this study that protocols using public-key cryptosystems and using conventional encryption algorithms are strikingly similar. The number of protocol messages exchanged is very comparable, the public-key system having a noticeable advantage only in the case of signed communications. As in many network applications of computers, caching is important to reduce transactions with lookup servers; this is particularly so with the public-key system. In that system we noticed also that there was a requirement for encryption of public data (the authentication server's database) in order to ensure its integrity. A consequence of the similarity of protocols is that any helpful tricks for the conventional system have analogs in the public-key system, though they may not be needed. Because of this, there may be scope for hybrid systems in which a public-key method

may be used to establish an authenticated connection to be used conventionally. The intrinsic security requirements of a public-key authentication server are easier to meet than those of a conventional one, but a complete evaluation of the system problems in implementing such a server in a real system, and the need to retain a secure record of old public keys to guarantee future correct arbitration of old signatures may minimize this advantage. We conclude that the choice of technique should be based on the economy and cryptographic strength of the encryption techniques themselves, rather than for their effects on protocol complexity.

Finally, protocols such as those developed here are prone to extremely subtle errors that are unlikely to be detected in normal operation. The need for techniques to verify the correctness of such protocols is great, and we encourage those interested in such problems to consider this area.

*Acknowledgments.* We are indebted to a number of people who have read drafts of this paper and made careful and helpful comments, notably: Peter Denning, Stockton Gaines, Jim Gray, Steve Kent, Gerry Popek, Ron Rivest, Jerry Saltzer, and Robin Walker.

Received September 1977; revised April 1978; final revision May 1978

**References**

1. Branstad, D. Security aspects of computer networks, Proc. AIAA Comptr. Network Syst. Conf., April 1973, paper 73-427.
2. Branstad, D. Encryption protection in computer data communications. Proc. Fourth Data Communications Symp., Oct. 1975, pp. 8.1–8.7 (available from ACM, New York).
3. Diffie, W., and Hellman, M. Multiuser Cryptographic Techniques, Proc AFIPS 1976 NCC, AFIPS Press, Montvale, N.J., pp. 109–112.
4. Feistel, H. Cryptographic coding for data bank privacy. Res. Rep. RC2827, IBM T.J. Watson Res. Ctr., Yorktown Heights, N.Y., March 1970.
5. Kent, S. Encryption-based protection protocols for interactive user-computer communication, M.S. Th., EECS Dept., M.I.T., 1976; also available as Tech. Rep. 162, Lab. for Comptr. Sci., M.I.T., Cambridge, Mass., 1976.
6. Kent, S. Encryption-based protection for interactive user/computer communication. Proc. Fifth Data Communication Symp., Sept. 1977, pp. 5-7–5-13 (available from ACM, New York).
7. National Bureau of Standards. Data Encryption Standard. Fed. Inform. Processing Standards Pub. 46, NBS, Washington, D.C., Jan. 1977.
8. Pohlig, S. Algebraic and combinatoric aspects of cryptography. Tech. Rep. No. 6602-1, Stanford Electron. Labs., Stanford, Calif., Oct. 1977.
9. Rivest, R.L., et al. A method for obtaining digital signatures and public-key cryptosystems. *Comm. ACM 21*, 2 (Feb. 1978), 120–126.

Programming Techniques

S. L. Graham
Editor

# A Linear Sieve Algorithm for Finding Prime Numbers

David Gries
Cornell University

Jayadev Misra
University of Texas at Austin

**A new algorithm is presented for finding all primes between 2 and *n*. The algorithm executes in time proportional to *n* (assuming that multiplication of integers not larger than *n* can be performed in unit time). The method has the same *arithmetic complexity* as the algorithm presented by Mairson [6]; however, our version is perhaps simpler and more elegant. It is also easily extended to find the prime factorization of *all* integers between 2 and *n* in time proportional to *n*.**

**Key Words and Phrases: primes, algorithms, data structures**

**CR Categories: 5.25, 5.24, 5.29**

## 1. Introduction

An algorithm is presented for finding all primes between 2 and $n$, for $n \geq 4$, that executes in time proportional to $n$. Like the sieve of Eratosthenes, it works by removing nonprimes from the set $\{2, \ldots, n\}$. Unlike the sieve of Eratosthenes, no attempt is ever made to remove a nonprime that was removed earlier; this allows us to develop a linear algorithm.

The algorithm deals with sets $S$ satisfying $S \subset \{2, \ldots, n\}$. Two operations will be required on such sets:

Permission to copy without fee all or part of this material is granted provided that the copies are not made or distributed for direct commercial advantage, the ACM copyright notice and the title of the publication and its date appear, and notice is given that copying is by permission of the Association for Computing Machinery. To copy otherwise, or to republish, requires a fee and/or specific permission.

This research was partially supported by the National Science Foundation under Grants DCR75-09842 and MCS76-22360.

Authors' addresses: D. Gries, Computer Science Department, Cornell University, Ithaca, NY 14853; J. Misra, Computer Science Department, University of Texas at Austin, Austin, TX 78712.

© 1978 ACM 0001-0782/78/1200–0999 $00.75