Public Key Cryptography

An introduction to a powerful cryptographic system for use on microcomputers.

John Smith
21505 Evalyn Ave.
Torrance, CA 90503

Cryptography, the art of concealing the meaning of messages, has been practiced for at least 3000 years. In the past few centuries, it has become an indispensable tool in the military affairs, diplomacy, and commerce of most major nations. During that time there have been many innovations, and cryptography has changed and grown to accommodate the increasingly complex needs of its users. Present techniques are very sophisticated and provide excellent message protection. Current developments in computer technology and information theory, however, are on the verge of revolutionizing cryptography. New kinds of cryptographic systems are emerging that have incredible properties, which appear to eliminate completely some problems that have plagued cryptography users for centuries. One of these new systems is public key cryptography.

In public key systems, as in most forms of cryptography, a piece of information called a key is used to transform a message into cryptic form. In conventional cryptography this key must be kept secret, for it can also be used to decrypt the message. In public key cryptography, however, a message remains secure even if its encryption key is publicly revealed. This unique feature gives public key systems great advantages over conventional systems.

This article deals with the theory and application of public key cryptography. It reviews the methods and problems of traditional cryptography and describes the remarkable concept and advantages of public keys. It also describes a real public key cryptosystem, showing examples of the encryption and decryption operations; and it attempts to clarify the concept of trap-door one-way functions, upon which public key systems are based.

Computers are essential for implementing many modern cryptosystems, including the one described here. Several BASIC-language programs (TRS-80) are included to illustrate algorithms used in this system. These can be used to experiment with the encryption, decryption, and derivation of small keys.

Conventional Cryptosystems

A cryptosystem must have two methods for transforming messages: a method of encryption, which renders messages unintelligible; and a method of decryption, for restoring them to their original forms. For simplicity, normal message text shall be called plaintext, and the encrypted form, ciphertext. Ciphertext messages may also be called cryptograms, or may just be called messages when it is clear that the encrypted form is meant.

To appreciate the significance of a public key system, we need to know some of the methods and problems of conventional cryptosystems. In a conventional system (see figure 1), a plaintext message is converted to a cryptogram by an encryptor and sent over a communication channel. While in transit, the cryptogram may be intercepted by someone other than the intended recipient. If it is encrypted well, it will be meaningless to the interceptor. At the receiving end, the cryptogram is converted back into plaintext by a decryptor. The encryptor and decryptor may be procedures executed by people or computers or may be specially constructed devices. In any case, they are both supplied with keys from a key source.

Cryptographic keys are analogous to the house and car keys we carry in our daily lives and serve a similar purpose. In many modern systems, each key is a string of digits. For example, keys defined by the Data Encryption Standard of the National Bureau of Standards consist of 64

TECMAR $1795.00

complete with controller & cartridge

FIRST FROM TECMAR

NEW REMOVABLE CARTRIDGE WINCHESTER

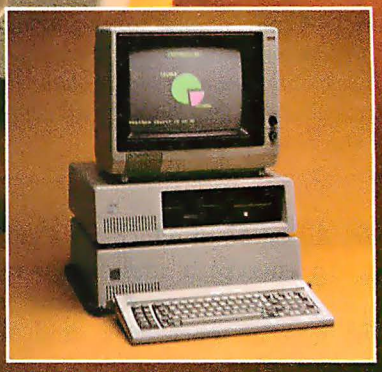
This is the breakthrough in storage that IBM PC people have been waiting for, as Tecmar engineering keeps you moving ahead.

- the new SyQuest **5 Megabyte removable** cartridge Winchester disk drive
- complete, easily installed in IBM PC or available in IBM-compatible Tecmar expansion chassis
- new Tecmar **superspeed** controller
- Tecmar disk sharing for up to 4 IBM PCs
- your best solution for **mass storage**, and the most sensible back-up system available.

We believe this is the future in storage because we have proved its **reliability** and its advantages. The new removable cartridge gives you unlimited memory at a lower price tag than the basic Winchester at comparable speed.

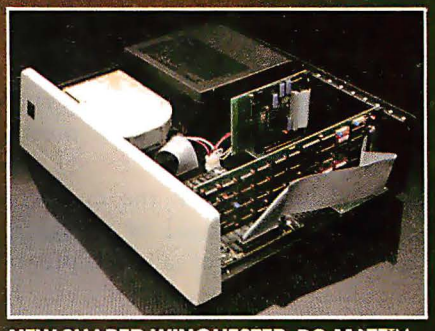
$1795 complete with controller & cartridge

AVAILABLE NOW AT YOUR TECMAR DEALER



TECMAR COMPATIBILITY, VERSATILITY, RELIABILITY, AFFORDABILITY, RESPONDABILITY

The first and only complete line of fully compatible expansion options for IBM PCs, including every type of disk drive



NEW SHARED WINCHESTER PC-MATE™

Our new GT subsystem upgrades our original with 3 times faster speed, sharing for up to 4 IBM PCs . . . Controller Board available for upgrade on trade-in.

PC-MATE™ FLOPPY

Controller Board will handle 5¼" and 8" disks. Winchesters can be installed in our floppy subsystem cabinet.

Write for new Tecmar Information Kit.

TECMAR

Personal Computer Products Division
23600 Mercantile Road
Cleveland, Ohio 44122
Phone 216-464-7410/Telex 241735

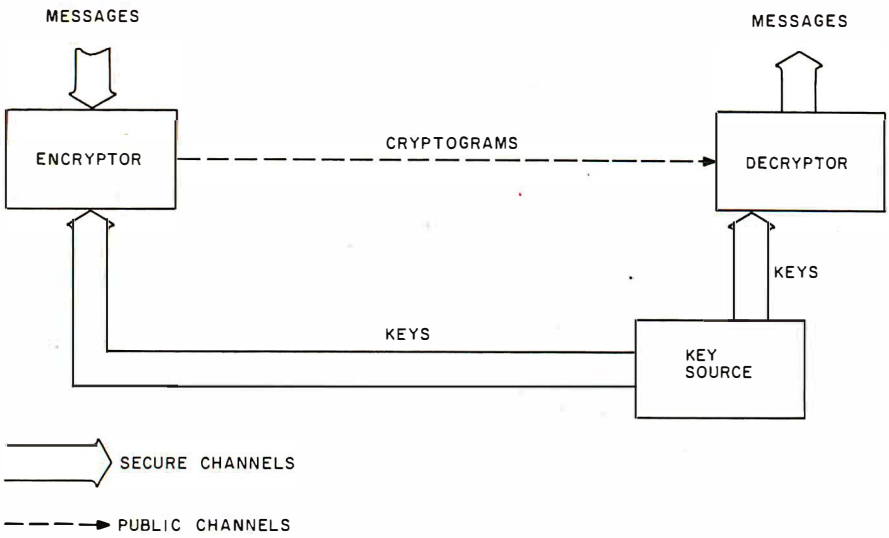


Figure 1: *A conventional cryptographic system. Encrypted messages (cryptograms) are sent over a public communication channel, while the keys needed for encryption and decryption are sent over secure channels, for example, via courier. The key source may be located at the encryptor or decryptor, in which case one of the secure channels is very short.*

binary digits, 56 of which are significant. To encrypt a message, a key and the message are somehow inserted into an encryptor, and the cryptogram that emerges is a jumble of characters that depends on both the message and the key. To decrypt the message, the correct key and the cryptogram are inserted into a decryptor, and the plaintext message emerges. In conventional systems, the correct key for decrypting a message is the same one used to encrypt it. Obviously, the keys used must be closely guarded secrets.

In a good system the number of possible keys should be very large, and decryption of any cryptogram should be possible with only very few of the keys, often with only one. These conditions make it impractical to try decrypting a message with one key after another until the one that reveals plaintext is found. The Data Encryption Standard provides more than 7×10^{16} keys (a 7 followed by 16 zeros), and there is some controversy over whether this number is sufficient!

The keys to be used are obtained from a key source, which selects them, perhaps randomly, from the large set of all usable keys. The key source may be located near the encryptor, near the decryptor, or elsewhere. But each key to be used must be made available to both the encryptor and the decryptor. Therein lies the most serious problem of conventional cryptosystems: some safe method must exist for distributing secret keys to the encryptor and the decryptor.

This problem is illustrated with a simple example: let's say you want to communicate privately with a friend named Mary. Many communication channels are available to you, none of which may be completely private: telephone, mail, and computer networks, for examples. You could send encrypted messages, but Mary could not read them without the keys. And you dare not send secret keys over these public channels. One of you must visit the other, so that you could agree on a key to use for future correspondence. But if your communication need was for only one private message exchange, it could be transacted during the visit, rendering the conventional cryptosystem unnecessary. Or if your communication need were immediate, a personal visit could cause an unacceptable delay. And if you need to communicate with several people, all the necessary visits could entail considerable expense. Most conventional cryptosystems, including the Data Encryption Standard system, have this problem. Public key cryptosystems, however, can avoid this problem entirely.

Public Key Systems

The concept of public keys may be one of the most significant cryptographic ideas of all time. A public key system has two kinds of keys: encryption keys and decryption keys. It may seem that having two kinds would make the key distribution problem worse, or at least no better. These keys, however, have remarkable, almost magical, properties:

- for each encryption key there is a decryption key, which is *not* the same as the encryption key
- it is feasible to compute a pair of keys, consisting of an encryption key and a corresponding decryption key
- it is not feasible to compute the decryption key from knowledge of the encryption key

Because of these properties, Mary and you can use a public key system to communicate privately without transmitting any secret keys. To set it up, you generate a pair of keys, and send the encryption key to Mary by any convenient means. It need not be kept secret. It can only encrypt messages—not decrypt them. Revealing it discloses nothing useful about the decryption key. Mary can use it to encrypt messages and send them to you. No one but you, however, can decrypt the messages (not even Mary!), as long as you do not reveal the decryption key. Figure 2 illustrates the flow of information in this situation, with Mary on the left and you on the right. To allow you to send private messages to her, Mary must similarly create a pair of keys, and send her encryption key to you.

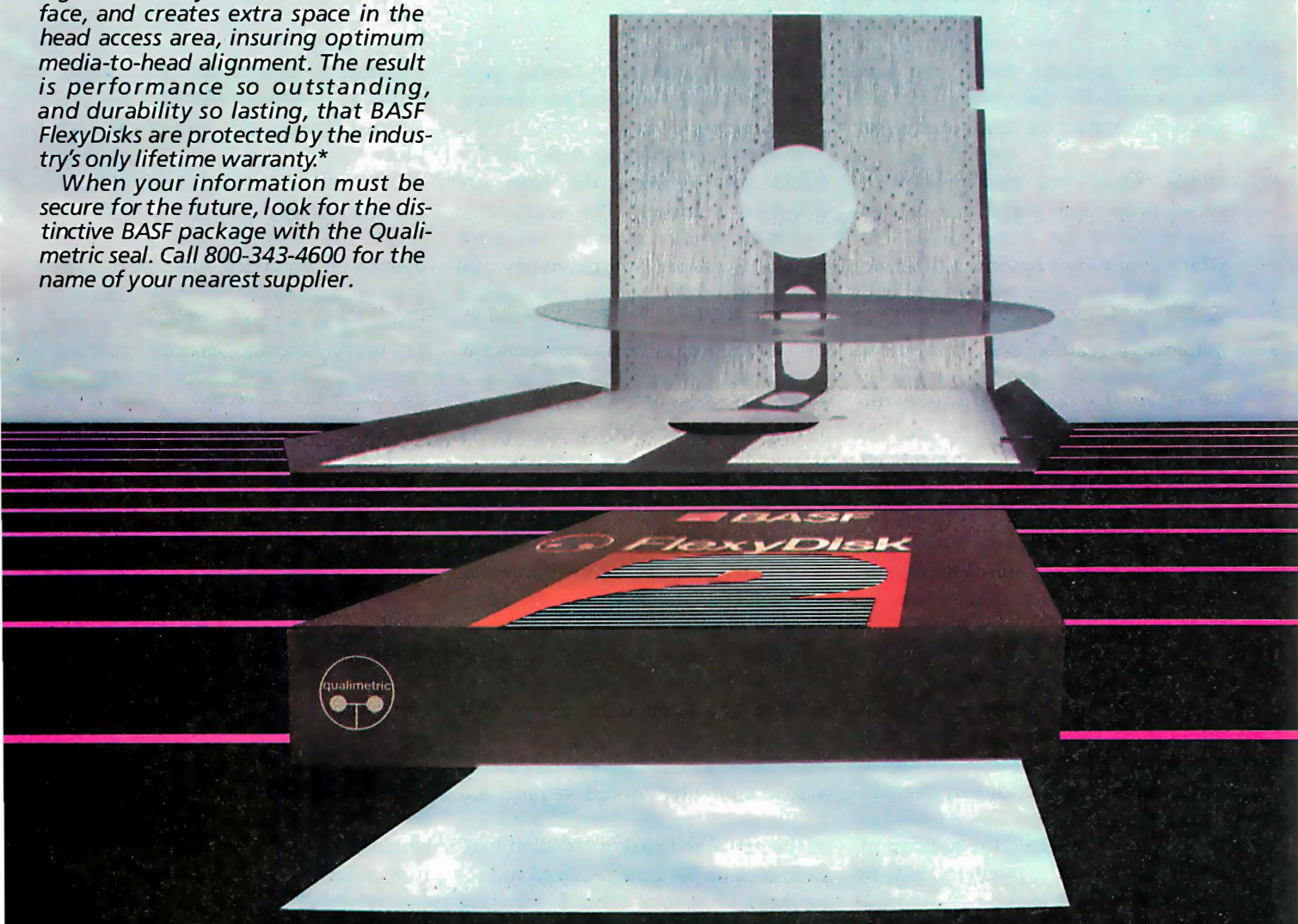
You can also go a step further. Since your encryption key need not be kept secret, you can make it public, for example, by placing it in a computer network public file. Once you have done so, anyone who wants to send you a private message can look up your public key and use it to

BASF QUALIMETRIC™
INSURING A TOMORROW FOR TODAY'S INFORMATION.

The BASF Qualimetric standard is a dramatic new international standard of quality in magnetic media...insurance that your most vital information will be secure for tomorrow when you enter it on BASF FlexyDisks® today.

*The Qualimetric standard reflects a continuing BASF commitment to perfection...a process which begins with materials selection and inspection, and continues through coating, polishing, lubricating, testing, and 100% error-free certification. Built into our FlexyDisk jacket is a unique two-piece liner. This BASF feature traps damaging debris away from the media surface, and creates extra space in the head access area, insuring optimum media-to-head alignment. The result is performance so outstanding, and durability so lasting, that BASF FlexyDisks are protected by the industry's only lifetime warranty.**

When your information must be secure for the future, look for the distinctive BASF package with the Qualimetric seal. Call 800-343-4600 for the name of your nearest supplier.



ENTER TOMORROW ON BASF TODAY

BASF

*Contact BASF for warranty details.

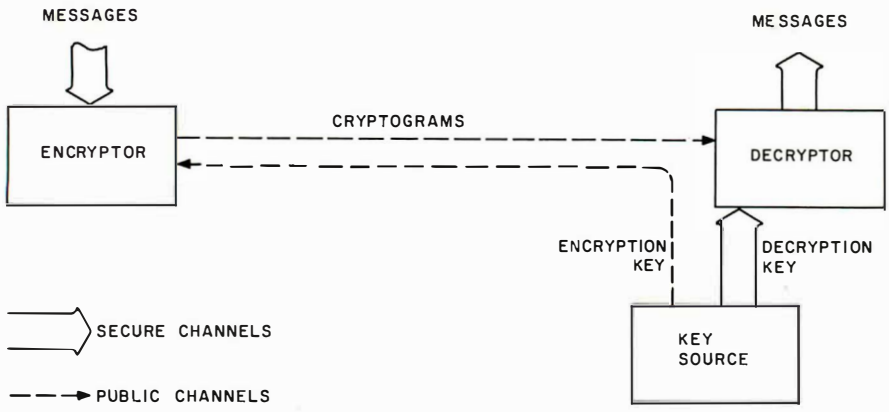


Figure 2: *A public key cryptographic system. Encryption keys can be safely sent over the ordinary communication channel because the information they contain cannot be used to decrypt messages. Decryption keys are created near the decryptor and are not sent anywhere else. Each person who expects to receive encrypted messages creates a key for encryption and a corresponding key for decryption and sends the encryption key to those who will originate the messages.*

Key Length (digits)	Factoring Time
50	3.9 hours
100	74 years
150	1.0 million years
200	3.8 billion years
250	5.9 trillion years

Table 1: *The time required to break the RSA public key system by factoring the key, for several different key lengths. These factoring times assume one computer operation per microsecond.*

encrypt a message. Since you need not transmit the decryption key, and since it cannot be computed from your public key, the message is secure. Only you can decrypt it. Other people can place their encryption keys in the same public file, which would thus become a directory of public keys. Any two people with directory entries could then communicate privately, even if they had no previous contact. It would be necessary, however, to protect the keys in such a file so that no one could change someone else's encryption key, for example, by substituting another encryption key. Fortunately, there is a way to protect the keys themselves with a public key cryptosystem, but that is another topic.

The RSA Cryptosystem

Now that the general concepts of public key cryptography have been examined, the next problem is how to design an actual working system. Indeed, when Whitfield Diffie and Martin Hellman conceived the basic properties of this cryptosystem in 1976, no one knew how to make a system that could employ them. The situation was similar to that of space travel in 1950. It was conceivable, but no one had accomplished it. In 1977, three researchers at the Massachusetts Institute of Technology, Ron Rivest, Adi Shamir, and Len Adleman, published an elegant method for creating and using public keys.

In the Rivest-Shamir-Adleman (or RSA) cryptosystem, the keys are 200-digit numbers. The encryption key is the product of two secret prime numbers, having approximately 100 digits each, selected by the person creating the keys. The corresponding decryption key is computed from the same two prime numbers, using a nonsecret formula.

Anyone who knows the secret prime numbers can compute the decryption key, but the primes are hidden because only their product, the encryption key, is revealed. Of course, the primes may be discovered by factoring the key, but factoring such a number is about as easy as traveling to Alpha Centauri, especially if the person who constructs the number has done it in a way that discourages factoring. Rivest, Shamir, and Adleman estimated that a fast computer would require 3.8 billion years (nearly the estimated age of the earth) to factor a 200-digit key. Estimates of the time required to factor keys of several other lengths are shown in table 1.

Before encryption, a message is converted into a string of numbers. This step is common in cryptosystems, as it is in computers and communication systems. Next, the message is subdivided into blocks, much as computer text files are subdivided into records or sectors. Each block contains the same number of digits, and is treated as one large number during encryption. To encrypt the message, an arithmetic operation involving the encryption key is performed on each block, resulting in a cryptogram containing as many blocks as the original message. The arithmetic operation, described below, is the same for all blocks. To decrypt, the inverse arithmetic operation, which requires the decryption key, is performed on each block of the cryptogram. The result is the original message in its numerical form.

As you can imagine, it would be cumbersome to illustrate these operations with 200-digit numbers, so the detailed descriptions below use small keys and messages; otherwise, the operations shown are the same as those used in a full-size RSA system. Also, the encryption method described here is actually a subset of the original RSA method. This modification, which is due to Donald Knuth (see reference 3), uses the basic RSA technique, while lessening somewhat the number of computations involved. (For more detailed information, the reader should refer to the original Rivest-Shamir-Adleman paper, shown as reference 5.)

BE CHALLENGED

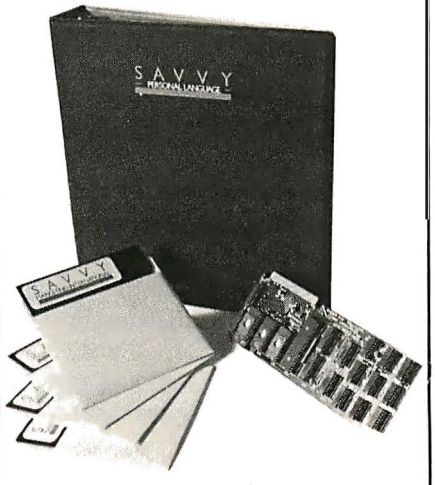
If you are a talented micro computer specialist, you follow the field closely, you know what's been done and what's expected. —

But you've never seen anything like

SAVVY™

Nobody has.

- A CO-PROCESSOR for the Apple II, with 64-Megabyte virtual Memory, 26 Decimal Digits of Precision, and Hardware Security.
- AN OPERATING SYSTEM that automatically Loads, Links, Overlays, and Executes Application Programs.
- A DATABASE MANAGER that automatically Blocks, Allocates, Opens, and Closes Data Sets.
- MACHINE INTELLIGENCE that automatically Resolves: Program Names, Item Names, Data Types, File Structures, Folder Names, and Instructions to the Robot Programmer™.
- A COMPILER that Produces Self-Loading, Self-Relocating, Serially Reusable Code.
- A PROGRAMMING LANGUAGE that uses Plain-text Language and Machine Intelligence.
- AN ASSOCIATIVE NETWORK in which Machine Intelligence, Operating System, Compiler, Database Manager, Programming Language, and Application Programs are all interconnected in a 20,000 name Associative Memory.



SAVVY can do things its developers haven't thought of. SAVVY may do things *you think of first.*

Excalibur wants to hear about your SAVVY project . . . and if it's really interesting (that's the challenge) we'll give you recognition and a commercial boost in national advertising.

To own SAVVY for $950.00 contact your local Apple dealer.

To accept the challenge, contact Excalibur, the developers of SAVVY, for detailed information on this remarkable system.

Excalibur Technologies Corp.
Albuquerque, New Mexico
(505) 242-3333

SAVVY Marketing International markets and distributes SAVVY the Personal Language™ System.

TM - SAVVY, Robot Programmer: Excalibur Technologies Corp.

TM - Personal Language is a trademark of SAVVY Marketing International.

Excalibur
TECHNOLOGIES CORPORATION

Circle 171 on inquiry card.

Arithmetic with a Modulus

The Rivest-Shamir-Adleman cryptography system uses arithmetic modulo n in encoding, decoding, and key selection. Because arithmetic modulo n is almost the same as ordinary arithmetic, it is easy to use.

To add or multiply modulo n, first add or multiply in the usual way. Then divide the result by n, and use the remainder for the final answer. For example, in arithmetic modulo 5, 3 + 4 = 2, because 3 + 4 is ordinarily 7, and 7 divided by 5 leaves a remainder of 2. This equation is usually written

$$(3 + 4) \bmod 5 = 2$$

where the notation "mod 5" indicates that arithmetic modulo 5 is being performed. Using this notation:

$$(4 \times 4) \bmod 5 = 1$$

since 4×4 = 16, and 16 divided by 5 leaves a remainder of 1.

The number n is called the modulus, and may be any positive integer. All answers in arithmetic modulo n are smaller than n, but are never negative. For example, when n is 5, every correct answer is 0, 1, 2, 3, or 4. If the initial result of addition or multiplication is less than n, the division step is unnecessary.

When performing a chain of operations, such as

$$(2 \times 3 \times 4) \bmod 5 = 4$$

the division step may be performed after each operation or at the end. The answer will be the same. When performing a chain of multiplications, it is best to perform the division step after every multiplication to keep the intermediate results from growing larger and larger. This is especially important where the intermediate results could overflow a computer's storage area.

Several common devices inherently perform arithmetic with a modulus. For example, most automobile odometers use a modulus of 100,000. If such an odometer reads 99,987 at the start of a 45-mile trip, it will read 32 at the destination; in the notation of arithmetic modulo n:

$$(99987 + 45) \bmod 100000 = 32$$

Computers are easily programmed to perform arithmetic modulo n. In BASIC, one extra statement is required for each arithmetic operation. For example, to calculate (A×B) mod n:

```
500  X = A*B
510  X = X - INT(X/N)*N
```

*Many interpreters allow placing both statements on the same line. INT(X/N) is the quotient that would result from division of X by N; INT(X/N)*N is the quotient times the divisor; and X - INT(X/N)*N is the remainder.*

In this article, an encryption operation is described that requires that a number be cubed modulo n. This BASIC subroutine computes B =(A³) mod n:

```
500 REM   COMPUTE B = (A*A*A)
    MOD N
510 B = A*A
520 B = B - INT(B/N)*N:
    REM   MOD N
530 B = B*A                :
    REM   (A*A)*A
540 B = B - INT(B/N)*N:
    REM   MOD N
550 RETURN
```

When multiplying integers, the number of digits in the result is usually the sum of the numbers of digits in the operands. If the result has more digits than the interpreter uses in its variables, the computed result will not be exact. Use double-precision variables, if they are available. Exact results will be obtained if the number of digits in the modulus is no more than half the number of digits used by the interpreter, and all operands are smaller than the modulus, which is usually the case.

How to Encrypt

While the encryption and decryption operations are normally performed by a computer program, I will describe them as if you were performing them by hand. Normally, the only manual operation required is entering the message to be encrypted.

Suppose you wish to encrypt the message

MARY HAD A LITTLE LAMB.

Once entered into a computer, the message will be in numerical form, frequently in ASCII (American Standard Code for Information Interchange). In ASCII, this message is

77 65 82 89 32 72 65 68 32
65 32 76 73 84 84 76 69 32
76 65 77 66 46

This is not yet encrypted, of course. It is merely written as a computer might represent it (all the numbers in this article are decimal). Group the message into blocks with six digits each:

776582 893272 656832 653276
738484 766932 766577 664600

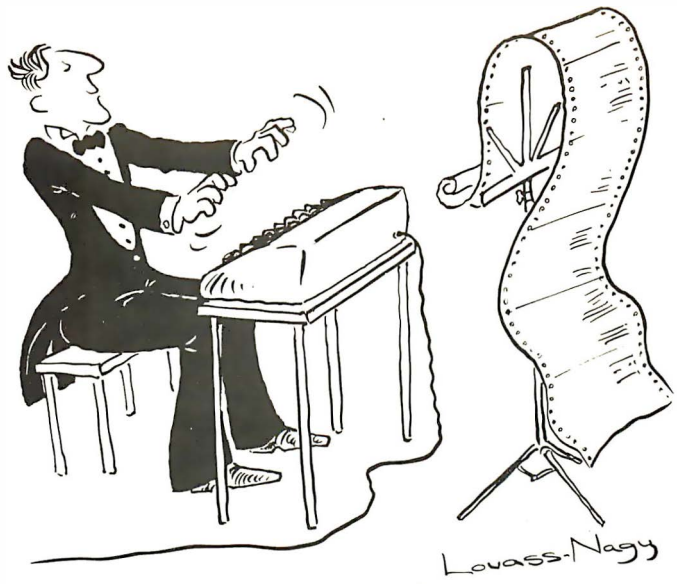
Each block except the last consists of three consecutive characters from the ASCII representation above. The last block consists of the last two characters plus two zeros added at the right to make the final block as long as the rest. Digits added for this purpose may have any value.

Suppose that the encryption key, usually called *n*, is 94815109. This is the product of two prime numbers. To encrypt the message, treat each block as a number, and cube it modulo *n* (see the text box "Arithmetic with a Modulus"). For example, to encrypt the first block of the message:

$$(776582 \times 776582 \times 776582) \bmod 94815109 = 71611947$$

Performing the cubing operation on all eight blocks produces the cryptogram

71611947 48484364 03944704
03741778 61544362 35331577
88278091 50439554



The Well-Tempered Cross-Assembler

Before Johann Sebastian Bach developed a new method of tuning, you had to change instruments practically every time you wanted to change keys. Very difficult.

Before Avocet introduced its family of cross-assemblers, developing micro-processor software was much the same. You needed a separate development system for practically every type of processor. Very difficult and very expensive.

But with Avocet's cross-assemblers, a single computer can develop software for virtually any microprocessor! Does that put us in a league with Bach? You decide.

Development Tools That Work

Avocet cross-assemblers are fast, reliable and user-proven in over 3 years of actual use. Ask NASA, IBM, XEROX or the hundreds of other organizations that use them. Every time you see a new microprocessor-based product, there's a good chance it was developed with Avocet cross-assemblers.

Avocet cross-assemblers are easy to use. They run on any computer with CP/M* and process assembly language for the most popular microprocessor families.

XASM05	6805	$200 each
XASM09	6809	
XASM18	1802	
XASM48	8048/8041	
XASM51	8051	
XASM65	6502	
XASM68	6800/01	
XASMF8	F8/3870	
XASMZ8	Z8	
XASM400	COP400	
XASM75	NEC 7500	$500

(Coming soon: XASM68K .... 68000)

Turn Your Computer Into A Complete Development System

Of course, there's more. Avocet has the tools you need from start to finish to enter, assemble and test your software and finally cast it in EPROM:

Text Editor VEDIT -- full-screen text editor by CompuView. Makes source code entry a snap. Full-screen text editing, plus TECO-like macro facility for repetitive tasks. Pre-configured for over 40 terminals and personal computers as well as in user-configurable form.

CP/M-80 version $150
CP/M-86 or MDOS version $195
(when ordered with any Avocet product)

In-Circuit Emulators -- MICE In-Circuit Emulator by Microtek. Full capability emulation in a compact, inexpensive device. Accepts high-level ASCII commands through RS 232 serial interface. Down-loads programs generated by Avocet cross-assemblers .. examine and modify memory and registers, access I/O ports and control program execution in single instruction and single-cycle modes. Forward and backward tracing for up to 256 qualified cycles ... Assembly/Disassembly commands with symbolic labels make it easy to modify the program under test.

MICE-I versions for 6502, 8048, 8085, NSC 800 and Z-80 ..... $1,795 each.

MICE-II versions with 2K trace and 32K program memory, plus real-time emulation and hardware break points for 6052, 6809, 6800, 8085 and 8086/8088 .... $3,995.

(6805 and 8051 versions available starting second quarter)

ROM Simulator -- ROMSIM by Inner Access eliminates need to erase and reprogram EPROM. Installed in an S-100 host, ROMSIM substitutes RAM for EPROM in external target system. 16K memory can be configured to simulate the 2708, 2758, 2716, 2516, 2732, 2532, 2764, 2564 in either byte or word organization. Avocet's configurable driver makes loading of HEX or COM files fast and easy.

From $495 depending on cabling and RAM installed.

EPROM Programmer -- Model 7128 EPROM Programmer by GTek programs most EPROMS without the need for personality modules. Self-contained power supply ... accepts ASCII commands and data from any computer through RS 232 serial interface. Cross-assembler hex object files can be down-loaded directly. Commands include verify and read, as well as partial programming.

PROM types supported: 2508, 2758, 2516, 2716, 2532, 2732, 2732A, 27C32, MCM8766, 2564, 2764, 27C64, 27128, 8748, 8741, 8749, 8742, 8751, 8755, plus Seeq and Xicor EEPROMS.

(Upgrade kits will be available for new PROM types as they are introduced.)

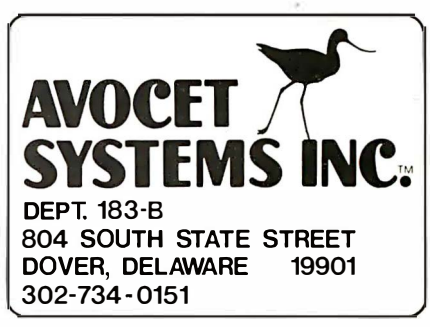
Programmer $389
Options include:
Software Driver Package ..... $ 30
RS 232 Cable ..... $ 30
8748 family socket adaptor ..... $ 98
8751 family socket adaptor $174

Call Us

If you're thinking about development systems, call us for some straight talk. If we don't have what you need, we'll help you find out who does. If you like, we'll even talk about Bach.

VISA and Mastercard accepted. All popular disc formats now available -- please specify. Prices do not include shipping and handling -- call for exact quotes. OEM INQUIRIES INVITED.

*Trademark of Digital Research.



Circle 45 on inquiry card.

Arithmetic modulo n is a fundamental part of the RSA system. It is also used in decryption and creating keys. Most of us have used arithmetic modulo n, although perhaps we didn't call it that. For instance, arithmetic modulo 12 is frequently used in calculations related to keeping time. The text box "Arithmetic with a Modulus" reviews the mechanics.

Almost any method may be used to convert the text to numbers. It would have worked just as well to use A=1, B=2, . . . Z=26, but the ASCII code is already in wide use, and it includes numbers for spaces and punctuation. The block length should be almost equal to the key length, because making it long minimizes the number of blocks per message. When considered as a number, however, no block should be as large as the key. For the above key, no block should be larger than 94815108. Making the block length slightly less than the key length ensures that this requirement is met. Of course, with full-length keys, there will be about 100 characters per block.

Listing 1 is a BASIC program that uses the above key to encrypt a line of text. Two lines of the program (670 and 680) perform the encryption. The rest deal with input, formatting, and printing. If desired, the encryption key in line 220 may be changed; use a key with seven or eight digits, or reduce the number of characters per block (line 210).

The programs in listings 1 through 4 were written for the TRS-80 BASIC interpreter, which is capable of 16-digit precision. They may be adapted for use with other interpreters, and I have tried to structure and annotate them well enough to make them easy to modify.

How to Decrypt

Since the RSA system is a public key system, the decryption key, usually called d, differs from the public encryption key. For the above encryption key, d is 63196467. Knowing the value of d, you can decrypt the message by raising each cryptogram block to the power d, modulo n. That is, if a cryptogram block is C, you must compute (C^d) mod n. For example, to decrypt the first block of the above cryptogram:

$$(71611947^{63196467}) \bmod 94815109 = 776582$$

converts this block back to the first three ASCII codes of the original message. Each of the remaining blocks is decrypted in the same way.

Fortunately, raising a number to a large power does not require performing a comparable number of multiplications. One efficient algorithm is a variation of the "Russian Peasant Method" of multiplication (see reference 4). It computes $M = (C^d) \bmod n$, as follows:

1. Let $M = 1$.
2. If d is odd, let $M = (M \times C) \bmod n$.
3. Let $C = (C \times C) \bmod n$.
4. Let $d =$ integer part of $d/2$.
5. If d is not zero, repeat from step 2; otherwise, terminate with M as the answer.

To raise a number to the power 63196467, this algorithm executes its loop (steps 2 through 5) 26 times. It is employed as a subroutine in the BASIC-language decryption program of listing 2. Line 200 contains the keys, which may be changed, if desired. Lines 340 through 380 execute the algorithm.

Text continued on page 210

Listing 1: *A program in BASIC (TRS-80) to demonstrate the encryption process described in the text. Lines 670-680 perform the encryption. When the program prompts you, type the text to be encrypted. The program will then print the text in numerical form, followed by the cryptogram. Use uppercase letters only.*

```
100 '=====================================================
110 ' ENCRYPT MESSAGES, USING A MINIATURE VERSION OF THE
120 ' RIVEST-SHAMIR-ADLEMAN PUBLIC KEY CRYPTOSYSTEM.
130 '
140 ' PROMPT FOR THE MESSAGE TO BE ENCRYPTED, PRINT THE
150 ' NUMERIC FORM OF THE MESSAGE, AND PRINT THE CRYPTOGRAM.
160 '=====================================================
170 ' DEFINE PARAMETERS.
180 '
190 DEFDBL C,M,N                    ' C, M, AND N HAVE 16 DIGITS
200 DIM M(100)                      ' MESSAGE BLOCKS
210 CHRS = 3                        ' CHARACTERS PER BLOCK
220 N = 94815109                    ' ENCRYPTION KEY, OR MODULUS
230 '-----------------------------------------------------
240 ' GET THE MESSAGE FROM THE USER.
250 '
260 PRINT : M$ = ""
270 INPUT "MESSAGE"; M$             ' MESSAGE FOR ENCRYPTION
280 IF M$ = "" THEN END             ' STOP IF NOTHING IS ENTERED
290 PRINT
300 '-----------------------------------------------------
310 ' ADD ZEROS TO MESSAGE, IF NECESSARY, TO MAKE ITS LENGTH
320 ' A MULTIPLE OF THREE (AN EVEN NUMBER OF BLOCKS).
330 '
340 L = LEN(M$)                     ' LENGTH OF MESSAGE
350 Q = INT(L/CHRS)                 ' NUMBER OF COMPLETE BLOCKS
360 R = L - Q * CHRS                ' LENGTH OF PARTIAL BLOCK
370 IF R > 0 THEN M$ = M$ + CHR$(0) : GOTO 340 ' ADD A ZERO?
380 '-----------------------------------------------------
390 ' CONVERT THE MESSAGE TO NUMERIC FORM, AND PRINT IT.
400 '
410 FOR I=0 TO Q-1                  ' I IS THE BLOCK NUMBER
420   M(I) = 0                      ' CONVERT BLOCK I TO NUMERIC
430   FOR J=1 TO CHRS                   ' FOR EACH CHAR IN BLOCK
440     A = ASC(MID$(M$,3*I+J,1))   ' CONVERT TO NUMBER
```

Listing 1 continued on page 208

Osborne™ brings you the comparison IBM® and Apple® don't want you to see.

Other computer companies dazzle buyers with an array of options and add-ons that makes the final price hard to determine and makes the computer hard to buy, complex to assemble, and very difficult to carry.

We believe in making personal computers that are easy to learn and use. And that starts with making computers easy to *buy*.

The Osborne 1™ Personal Business Computer. One simple price, $1795, buys it all.

And it all comes in a portable case you can take with you wherever you work. Because once you go to work with an Osborne, you won't want to work any other way.

For your nearest dealer, call (in California) 800 772-3545, ext. 905; (outside California) call 800 227-1617, ext. 905.

OSBORNE™
COMPUTER CORPORATION

	OSBORNE 1™	IBM PERSONAL®	APPLE II®
Computer with 64K RAM, two floppy drives[A], keyboard and CRT:	$1795	$3240[B]	$3120[C]
Serial communications:	INCLUDED	EXTRA COST	EXTRA COST
Modem Connection:	INCLUDED	EXTRA COST	EXTRA COST
IEEE 488 Instrument communications:	INCLUDED	EXTRA COST	EXTRA COST
BASIC interpreter[D]:	INCLUDED	INCLUDED	INCLUDED
Business BASIC[E]:	INCLUDED	EXTRA COST	EXTRA COST
CP/M® Control Program:	INCLUDED	EXTRA COST	F (see below)
Word Processing[G]:	INCLUDED	EXTRA COST	EXTRA COST
Electronic Spreadsheet[H]:	INCLUDED	EXTRA COST	EXTRA COST
Carrying Case:	INCLUDED	EXTRA COST	EXTRA COST
TOTAL PRICE[I]:	**$1795**	**$4000–4700**	**$4000–4700**

A. The Osborne 1™ includes two built-in 100K byte floppy disk drives. The IBM® and APPLE II® drives provide approximately 160K bytes of storage. **B.** From the IBM Product Center Personal Computer Price Schedule. **C.** From the Apple Computer Suggested Retail Price List. **D.** The Osborne includes MBASIC® from Microsoft. **E.** The Osborne includes CBASIC®, a business-oriented BASIC language from Digital Research.™ **F.** The Osborne includes CP/M®, the industry-standard control program from Digital Research. The list of software packages which will run with CP/M is considerable. IBM offers CP/M 86 (a version of CP/M) at extra cost. There are optional hardware systems which allow the Apple II to run CP/M; the Apple II control program is highly comparable to CP/M. **G.** The Osborne includes WORDSTAR® word processing with MAILMERGE®—products of MicroPro™ International. **H.** The Osborne includes SUPERCALC™, the electronic spreadsheet system from Sorcim Corporation. **I.** Exact price comparisons cannot be presented, because the software and hardware options chosen to create the "equivalent" of the Osborne 1 Personal Business Computer vary in price. The range indicated was computed using price lists from IBM and Apple. Documentation of the computations are available on request from Osborne Computer Corporation. Trademarks: OSBORNE 1: Osborne Computer Corporation; SUPERCALC: Sorcim Corporation; Digital Research: Digital Research, Inc.; Registered Trademarks: WORDSTAR, MAILMERGE: MicroPro International Corporation of San Rafael, CA; MBASIC: Microsoft; CBASIC, CP/M: Digital Research, Inc.; IBM: IBM Corporation; Apple, Apple II: Apple Computer Corporation.

Listing 1 continued:

```
450     M(I) = M(I) * 100              ' SHIFT BLOCK LEFT
460     M(I) = M(I)+ A                 ' ADD THE CHARACTER
470   NEXT J
480   PRINT M(I);                      ' PRINT THE BLOCK
490 NEXT I                             ' DO THE NEXT BLOCK
500 PRINT : PRINT
510 '-------------------------------------------------------
520 ' ENCRYPT THE MESSAGE, AND PRINT THE CRYPTOGRAM.
530 '
540 PRINT "CRYPTOGRAM:" : PRINT
550 FOR I=0 TO Q-1                     ' I IS THE BLOCK NUMBER
560   M = M(I)
570   GOSUB 670                        ' ENCRYPT THE BLOCK
580   PRINT C;                         ' PRINT IT
590 NEXT I                             ' DO THE NEXT ONE
600 PRINT
610 '-------------------------------------------------------
620 GOTO 260                           ' RUN THE PROGRAM AGAIN
630 '-------------------------------------------------------
640 ' SUBROUTINE.  ENCRYPT ONE MESSAGE BLOCK.
650 ' COMPUTE C = (M^3) MOD N.
660 '
670 C = M * M : C = C - INT(C/N) * N     ' (M * M) MOD N
680 C = C * M : C = C - INT(C/N) * N     ' (M * M * M) MOD N
690 RETURN
700 '=======================================================
```

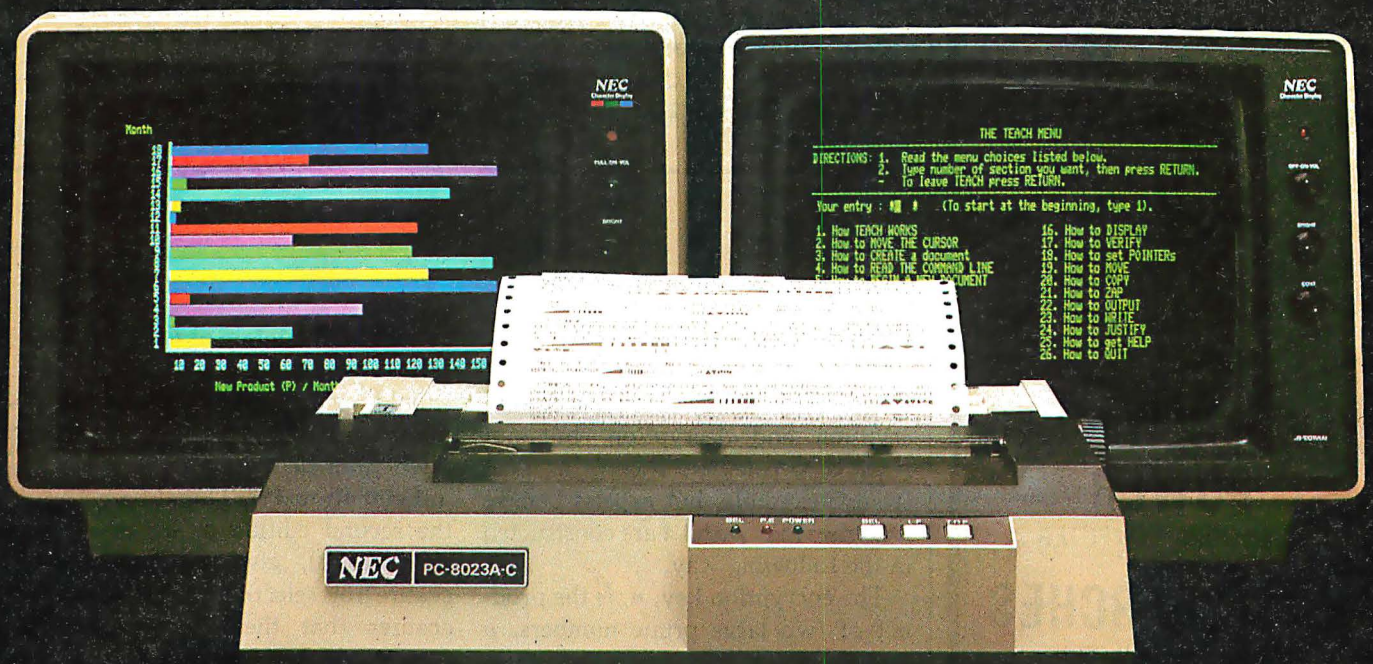
Listing 2: *A program in BASIC (TRS-80) to demonstrate the decryption process described in the text. Lines 340-390 decrypt one block of a cryptogram by raising it to a power. The program asks for a cryptogram block to be decrypted. Several seconds later, it prints the decrypted characters in ASCII. If you enter 0, the program will terminate.*

```
100 '=======================================================
110 ' DECRYPT MESSAGES, USING A MINIATURE VERSION OF THE
120 ' RIVEST-SHAMIR-ADLEMAN PUBLIC KEY CRYPTOSYSTEM.
130 '
140 ' PROMPT FOR THE CRYPTOGRAM BLOCK TO BE DECRYPTED, AND
150 ' DECRYPT AND PRINT THE MESSAGE BLOCK, IN NUMERIC FORM.
160 '=======================================================
170 ' DEFINE PARAMETERS.
180 '
190 DEFDBL C,D,M,N                     ' DOUBLE PRECISION
200 N = 94815109 : D = 63196467        ' KEYS
210 '-------------------------------------------------------
220 ' MAIN PROGRAM LOOP.
230 '
240 INPUT "CRYPTOGRAM BLOCK"; C        ' USER ENTRY
250 IF C = 0 THEN END                  ' STOP IF NO ENTRY
260 GOSUB 340                          ' DECRYPT BLOCK
270 PRINT M                            ' MESSAGE BLOCK
280 GOTO 240                           ' REPEAT
290 '-------------------------------------------------------
300 ' SUBROUTINE.  DECRYPT C, CRYPTOGRAM BLOCK.
310 ' COMPUTE M = (C^D) MOD N.  USE MODIFIED RUSSIAN PEASANT
320 ' ALGORITHM (BYTE, OCTOBER 1981, PAGE 376).
330 '
```

Listing 2 continued on page 210

NEC's crisp, clear, high-performance JC1203 RGB color monitor, an industry standard. Also available, the JC1212 composite video version.

NEC's classic JB1201 green monitor, one of microcomputing's performance legends. Easy on the eye, and the checkbook.



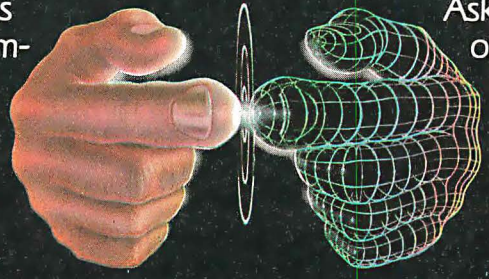
Our impressive new NEC dot matrix printer. Parallel interface, 100 cps, 2K buffer, pin or friction feed. Stunning performance and compatibility in the hottest new peripheral of the year.

Give your IBM system some NEC, and watch its performance soar.

Peripherals from NEC can make almost any computer system better.

Our sparkling new JC1203 color monitor is plug and pin compatible with the 16-color IBM® PC, and delivers the bright, sharp, clear, and stable screen image for which the entire NEC line has long been famous. Similar compatibility is available to owners of Apple II®, Radio Shack®, and Atari® computers, not to mention our own outstanding NEC PC-8000 series. Also available is a brand new, extremely low cost, NEC green monochrome monitor, the JB1260, perfect companion for an Osborne®, for instance.

Ask your dealer for a demonstration. Or write us at 1401 Estes Avenue, Elk Grove Village, IL 60007.



Productivity at your fingertips™

NEC

NEC Home Electronics (U.S.A.), Inc.
Personal Computer Division

Nippon Electric Co., Ltd., Tokyo, Japan

Listing 2 continued:

```
340 D1 = D : M = 1
350 IF D1/2 = INT(D1/2) GOTO 370          ' IF D1 IS EVEN, SKIP
360 M = M * C : M = M - INT(M/N) * N      ' M = (M * C) MOD N
370 C = C * C : C = C - INT(C/N) * N      ' C = (C * C) MOD N
380 D1 = INT(D1/2) : IF D1 > 0 GOTO 350
390 RETURN
400 '=============================================================
```

How to Derive Keys

Earlier, I said that it is feasible to derive a pair of keys, n and d, for encryption and decryption, but not feasible to calculate d from n. That seems incredible, but experts believe it is true when n and d are constructed in the following way.

The encryption key, n, is the product of two large prime numbers, p and q:

$$n = pq \qquad (1)$$

The decryption key, d, is calculated from p and q by

$$d = [\, 2(p-1)(q-1) + 1 \,]/3 \qquad (2)$$

Although n is made public, p and q remain secret. If n is sufficiently large, say 200 digits, it is practically impossible for anyone to factor it and discover the values of p and q; and without knowing p and q, it is equally difficult to compute d.

For the encryption and decryption examples given earlier, the keys were constructed as follows:

prime number, $p = 7151$
prime number, $q = 13259$
encryption key, $n = 7151 \times 13259$
$= 94815109$
decryption key, $d = (2 \times 7150 \times 13258 + 1)/3$
$= 63196467$

Because p and q may have 100 or more digits in an operational RSA system, their selection requires computer assistance. The following three restrictions apply to how they should be chosen. First, neither $p-1$ nor $q-1$ must be divisible by 3, or the decryption operation will not work correctly. Second, $p-1$ and $q-1$ should both contain at least one large prime factor. Third, the ratio p/q should not approximate a simple fraction, e.g., ⅔, ¾, etc. These last two restrictions help ensure that n will be difficult to factor. Donald Knuth, in the second edition of his book (see reference 3), gives a detailed procedure for selecting p and q, which ensures that these restrictions are met. While the procedure described is for constructing 250-digit keys, it is applicable to other key lengths.

Enough keys are available for everyone. The number of 250-digit keys constructible with Knuth's procedure is much greater than 10^{200}. For comparison, the number of atoms in the known universe is about 10^{80}.

To create a different pair of seven- or eight-digit keys, find primes p and q such that neither $p-1$ nor $q-1$ is divisible by 3, and the product $n=pq$ is a seven- or eight-digit number. Then calculate d from formula (2). Divisibility by 3 is easily checked by casting out 3s, and the BASIC programs described below are helpful in finding prime numbers.

How to Find Large Prime Numbers

To find a large prime number, select a random odd number of the required size and determine whether it is prime. If it is not, increase it (or decrease it) by 2 and try again, repeating until finding a prime. It is not necessary, however, to attempt to factor a number to determine whether it is prime.

To test whether a number n is prime, select any number greater than 1 and smaller than n, say x, and calculate

$$y = (x^{n-1}) \bmod n$$

If y is not equal to 1, n is not prime. But if $y = 1$, n *may* be prime, and further testing is required. Repeat the test using another value of x. If this test is performed with many different values of x, and if $y = 1$ for all the test cases, n is probably prime. Listing 3 is a BASIC program that uses 10 values of x to test a number for primality. If the program says the number is not prime, it is not prime. But if the program says the number is probably prime, there is a small chance that it is not.

What is the probability that this program will make an error? I don't know, but it illustrates a class of programs, some of which are very good. Knuth (reference 3, page 375) presents one that is slightly more complicated, for which the odds against an error are a million to one when 10 values of x are used for testing, and are a million million to one when 20 values are used. For serious work I would use the more complicated program, but the one presented here illustrates the process of testing without factoring—and it doesn't seem bad. It has not made an error in several hundred trials.

Listing 4 is a BASIC program that searches for a prime number using the same test method as the previous program. The program will begin with the number you enter and search downward until it finds a probable prime, which it will identify. If you enter 99999999, it will find the largest eight-digit prime. This program helps to find primes for constructing small keys like the ones above.

Listing 3: *A program in BASIC (TRS-80) to test whether a number is prime. This program demonstrates a primality test that does not attempt to factor the number being tested. For very large numbers, it is much faster than factoring.*

```
100 '=================================================
110 ' TEST WHETHER A NUMBER IS PRIME.
120 ' USE PROBABILISTIC TEST BASED ON FERMAT'S THEOREM.
130 ' SEE KNUTH, "SEMINUMERICAL ALGORITHMS".
140 '
150 ' PROMPT FOR NUMBER, TEST IT, AND PRONOUNCE VERDICT.
160 '=================================================
170 ' DEFINE PARAMETERS.
180 '
190 DEFDBL N,P,X,Y                  ' DOUBLE PRECISION
200 K = 10                          ' NUMBER OF TEST CASES
210 '-------------------------------------------------
220 ' GET A NUMBER TO BE TESTED.  CHECK THE SIZE.
230 '
240 PRINT
250 INPUT "NUMBER"; N               ' GET A NUMBER TO TEST
260 IF N < 3 THEN END
270 IF N > 99999999 THEN PRINT "TOO BIG" : GOTO 240
280 '-------------------------------------------------
290 ' DETERMINE WHETHER N IS PRIME.
300 '
310 PRINT "TEST NUMBER: ";
320 FOR I=1 TO K                    ' TEST CASES
330     X = 2 + INT((N-2)*RND(0))       ' TEST VALUE
340     PRINT X;
350     GOSUB 490                       ' PERFORM TEST
360     IF Y <> 1 GOTO 380              ' NOT PRIME?
370 NEXT I
380 PRINT : PRINT                   ' NOT PRIME IF Y <> 1
390 '-------------------------------------------------
400 ' PRINT THE VERDICT.
410 '
420 IF Y = 1  THEN PRINT N; "IS PROBABLY PRIME."
430 IF Y <> 1 THEN PRINT N; "IS NOT PRIME."
440 '-------------------------------------------------
450 GOTO 240                        ' RUN THE PROGRAM AGAIN
460 '-------------------------------------------------
470 ' SUBROUTINE.  COMPUTE Y = [X^(N-1)] MOD N.
480 '
490 Y = 1 : P = N-1
500 IF P/2 = INT(P/2) GOTO 520          ' IF P IS EVEN, SKIP
510 Y = Y * X : Y = Y - INT(Y/N) * N ' (Y * X) MOD N
520 X = X * X : X = X - INT(X/N) * N ' (X * X) MOD N
530 P = INT(P/2) : IF P > 0 GOTO 500
540 RETURN
550 '=================================================
```

One-Way Functions and Trap-Doors

Public key cryptosystems derive their unusual properties from mathematical functions called trap-door one-way functions, which are useful because they can act as ordinary functions or as one-way functions.

One-way functions are like one-way streets. The ordinary cube function, $B = A^3$, resembles a one-way function in that it is easier to calculate B, given A, than it is to calculate A, given B. The latter calculation, the cube-root function, is called the inverse of the cube function. The inverse of an automobile would convert smog to gasoline. A mathematical function is said to be one-way if it is much more difficult to compute the inverse than to compute the function itself. To qualify as a one-way function, the inverse must be very difficult to compute, even by machine. A function that could be computed in a few seconds, for which computing an inverse required thousands of years, would fit the definition.

To create a public key cryptosystem, a trap-door one-way function is used. It is easy to compute an inverse of a trap-door one-way function, but

DISC-LESS $1050

Sonics Micro Systems announces the commercial availability of S.D. Systems' . . .

First "Disc-Less" Micro Computer System. $1050

1. Replaces Floppy Disc Drives.
2. CP/M,MP/M, Oasis and Turbo-Dos compatible.
3. Transparent to operating system disc commands.
4. No moving parts, no alignment, no media failures.
5. Network ready.
6. FASTIIIIIIIIIIIIIIII

"Ram-Disc" $630

• Operating under CP/M the "Ram-Disc 128" functions as a fully compatible floppy drive replacement. Maximum single board configuration of 256K Bytes offers the equivalent capacity of 8" floppies. If more local disc image storage is necessary the Ramdisc system may be expanded to a full 40M Bytes.

• Whether operations require the "Ram-Disc 128" to operate as a floppy replacement or as a high speed data acquisition system is solely dependent on single system configuration.

• In real time data acquisition and subsequent processing applications the "Disc-less" system approach affords mini computer speed and versatility at micro computer prices.

"Rom-Disc" $289

• The "Rom-Disc-128" is a direct replacement for floppy disc drives used for the purpose of booting the CP/M operating system. Further the "Rom-Disc-128" is a direct replacement for floppy disc drives used to load and store applications programs.

• A total lack of sensitivity to the storage and handling parameters of standard floppy discs make the "Rom-Disc-128's" media virtually "immune" to familiar system failures.

• Under popular CP/M utilities the "Rom-Disc-128" appears as a simple disc drive.

• With CP/M configured in the S.D. Rom format, systems boot in less than 1/10 of a second.

• Equipped with a high speed RS-232 serial port the "Rom-Disc-128" will accommodate data transfer to and from the host.

• The "Rom-Disc-128" in conjunction with the previously described "Ram-Disc-128" provide true system portability and independence from floppy disc drives.

• Each "Rom-Disc-128" may be attached to a 256K Byte applications "personality module" allowing maximum system flexibility and personality. "Rom-Disc-128" because of its very nature offers "maximum" protection from software piracy.

"Turbo-Dos" $350

Z-80 CP/M compatible network ready Turbo-Dos in stock, ready for immediate delivery.

Versa Floppy II "With CP/M 3·0" $475

• Supports dual 5¼", dual 8" or both. "CP/M 3-0" included.

SONICS
MICRO SYSTEMS INC.
1500 N.W. 62 STREET
FORT LAUDERDALE, FL 33309
1-800-327-5567
IN FLORIDA CALL: 305-776-7177

at Sonics "We are Technology"

Circle 378 on inquiry card.

it can be very difficult to determine how. Computing an inverse can take millions of years because finding out how to do it can take that long. If the method is known, computing an inverse may take only a few seconds. This is a completely different situation than that created by a one-way function, for which there is no easy way to compute an inverse. When a trap-door one-way function is being constructed, the person constructing it has access to information, called trap-door information, that reveals how to compute inverses. Once the function is constructed, the trap-door information is hidden so well that it can take millions of years to find.

The Knuth modification of the RSA system encryption function, cubing a number modulo n, is a trap-door one-way function. Its inverse function is the cube root modulo n. In arithmetic modulo n, "cube root" is defined as in ordinary arithmetic: if B is the cube of A, then A is the cube root of B. Notice that this definition does not say how to compute cube roots (in either kind of arithmetic). If you know how to compute cube roots modulo n, you know how to decrypt messages. In modulo n arithmetic, the cube root of B is computed by raising B to some power d, modulo n. But knowing this doesn't help unless you know the value of d. And d can be computed by formula (2) if n has two factors (p and q), and $p-1$ and $q-1$ are not divisible by 3. If you construct the modulus, n, you know p and q, and can therefore calculate the value of d. Knowing d, you can compute cube roots; in other words, decrypt cryptograms. The values of p and q are hidden from other people by the difficulty of factoring n. They are deprived of the value of d, and therefore cannot compute cube roots. Hence, they cannot decrypt cryptograms created by cubing modulo n. In the RSA system, the value of d is the trap-door information that reveals how to compute inverses (cube roots). You might think of p and q as comprising a trap-door through which the value of d is obtained. Factoring n is analogous to finding the trap-door, but it is very difficult to do.

Listing 4: *A program in BASIC (TRS-80) that searches for a prime number. It illustrates the search technique and may be used to help construct small keys for the public key cryptosystem described in the text. Enter any number of eight digits or fewer, and the program will find a prime number that does not exceed the number entered.*

```
100 '=====================================================
110 ' FIND A PRIME NUMBER NO LARGER THAN THE NUMBER ENTERED.
120 ' USE PROBABILISTIC TEST BASED ON FERMAT'S THEOREM.
130 ' SEE KNUTH, "SEMINUMERICAL ALGORITHMS".
140 '=====================================================
150 ' DEFINE PARAMETERS.
160 '
170 DEFDBL N,P,X,Y                  ' DOUBLE PRECISION
180 K = 10                          ' NUMBER OF TEST CASES
190 '-----------------------------------------------------
200 ' GET A NUMBER TO BE TESTED.  CHECK THE SIZE.
210 '
220 PRINT
230 INPUT "NUMBER"; N               ' GET A NUMBER TO TEST
240 IF N < 3 THEN END               ' STOP IF SMALL NUMBER
250 IF N > 99999999 THEN PRINT "TOO BIG" : GOTO 220
260 '-----------------------------------------------------
270 ' DETERMINE WHETHER THE NUMBER ENTERED IS EVEN.
280 ' IF SO, SUBTRACT ONE.
290 '
300 IF N/2 = INT(N/2) THEN N = N - 1
310 '-----------------------------------------------------
320 ' PRINT N, THEN DETERMINE WHETHER IT IS PRIME.
330 '
340 PRINT N;
350 FOR I=1 TO K                            ' TEST CASES
360     X = 2 + INT((N-2)*RND(0))           ' TEST VALUE
370     GOSUB 520                           ' PERFORM TEST
380     IF Y <> 1 GOTO 400                  ' NOT PRIME?
390 NEXT I
400 REM
410 '-----------------------------------------------------
420 ' IF N IS PRIME, TERMINATE THE PROGRAM.  OTHERWISE,
430 ' DECREASE IT BY TWO, AND TRY AGAIN.
440 '
450 IF Y = 1 THEN PRINT "IS PROBABLY PRIME." : END
460 PRINT "NO." : N = N - 2 : GOTO 340
470 '-----------------------------------------------------
480 GOTO 220                        ' RUN THE PROGRAM AGAIN
490 '-----------------------------------------------------
500 ' SUBROUTINE.  COMPUTE Y = [X^(N-1)] MOD N.
510 '
520 Y = 1 : P = N-1
530 IF P/2 = INT(P/2) GOTO 550              ' IF P IS EVEN, SKIP
540 Y = Y * X : Y = Y - INT(Y/N) * N        ' (Y * X) MOD N
550 X = X * X : X = X - INT(X/N) * N        ' (X * X) MOD N
560 P = INT(P/2) : IF P > 0 GOTO 530
570 RETURN
580 '=====================================================
```

Other trap-door one-way functions undoubtedly exist, and these could be the foundations for other public key cryptosystems. For each of these systems, the same principles would apply. The creator of the system parameters would have access to certain trap-door information, which

GP-100A: US$389

COMMAND PERFORMANCE.

Seikosha gives you all the best features—including economy and super-clear graphics.

Unlike some graphic printers, Seikosha's new GP-100A Uni-Hammer Graphic Printer puts full dot addressable graphics at your command. The GP-100A lets you repeat a column of data as many times as needed with just one command. Software control enables double-width character output, and the positioning is both character and dot addressable. Designed for simple operation, it ranks among the most cost-efficient graphic printers on the market. Command performance technology that not only works for you now, but takes you well into the future.

Other valuable features:

- Graphics, regular and double width character modes can be intermixed on the same line.
- Automatic printing. When the text exceeds the maximum line length, there is no loss of data due to overflow.
- Self-test printing is a standard feature.
- Centronics type parallel interface.
- Paper width is adjustable up to 10 inches.
- Optional Interface: RS232C, IEEE488, apple II, etc.

Graphic Printer GP Series

Available at COMPUTERLAND and other fine stores in your area

Distributed by **AXIOM CORPORATION** 1014 Griswold Avenue San Fernando, Calif. 91340 Phone (213) 365-9521 TWX (910) 496-1746

Manufactured by **SEIKOSHA SYSTEM EQUIPMENT DIV.** 4-1-1 Taihei Sumida-ku Tokyo Japan. Phone: 03-623-8111 Telex: 262-2620

Circle 367 on inquiry card.

Editor's Note: *Recently, a software product became available that allows Z80 system owners to take advantage of the benefits offered by public key cryptography in their private correspondence. Called The Protector (from Standard Software of Randolph, Massachusetts; list price: $165), the new system uses a 77-digit key. On a 4-MHz Z80 microcomputer running under the CP/M operating system, message encryption and decryption take about one minute plus the necessary disk access time. The time needed to generate the encryption and decryption keys ranges from 15 minutes to 4 hours. The memory requirement is 38K bytes.*

Although the 77-digit key is much shorter than the 200-digit key proposed for the full-size Rivest-Shamir-Adleman system, the key may be more than adequate for most applications. The author of the system, Charles Merritt of PKS Inc., has received estimates of the time needed to break the system ranging from three uninterrupted days on a Cray-1 to one year.

*When asked about the people who were using the system, Mr. Merritt replied that he had not heard from any of them. Apparently, they also want to keep their identities secret. . . .***R. M.**

would reveal how to compute inverses. For everyone else, the trapdoor would be hidden, and for them the encryption function would be, in effect, a one-way function.

Is the RSA System Unbreakable?

Successfully analyzing a cryptosystem, and being able to read its cryptograms without authorization, is called breaking the system. Theoretically, the RSA system can be broken by a determined analyst. Factoring the encryption key, or modulus, would do the trick, for then the decryption key could be easily calculated from formula (2), after which any message could easily be decrypted. However, factoring a key of the recommended length and construction does not seem feasible. Knuth gives a procedure for constructing a 250-digit key and considers it inconceivable at this time that such a key could be factored. Experts acknowledge that a breakthrough in the art of factoring large numbers would render the RSA system worthless but consider such a breakthrough extremely unlikely. Apparently, factoring large numbers is not a new problem, but one that expert mathematicians have attacked for centuries, and it is known to be very difficult.

Another way to break the system is to determine the value of d without factoring n. Although you can approach this problem in several ways, experts believe that none of them are likely to be fruitful.

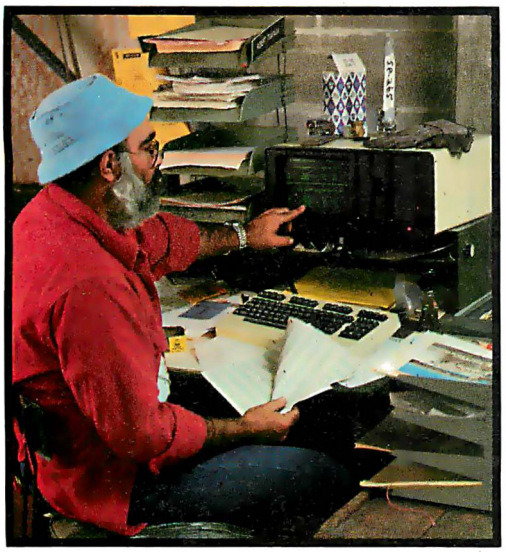
Yet another method of breaking the system is to learn how to compute cube roots modulo n without knowing the value of d. Less seems to be known about the difficulty of doing this than is known about the difficulty of factoring n. At this time, no one knows how to compute such cube roots in a reasonable time without knowing d.

Any new cryptosystem should be viewed with suspicion. The accepted method of demonstrating the adequacy of a new system is to subject it to prolonged, concerted attack by people with experience in breaking other systems. If the new system proves resistant to such an attack, it may tentatively be considered secure. The process of validation is continuing, but a fairly large number of preliminary studies done so far indicate that the system is quite secure.

Digital Signatures

Very closely related to public key cryptography is the concept of digital signatures. One problem with corresponding electronically, such as via a computer network, is that messages can be easily *forged*—you usually cannot be certain that the sender of a received message is actually the person claimed in the message. A public key cryptosystem, however, can be used to provide positive identification of any sender who has a public key

The new DMS-15. Today, a desktop powerhouse. Tomorrow, a HiNet™ network master.



As a stand-alone, the DMS-15 speeds word processing and accounting with its own 15MByte Winchester disk. As "master" for a HiNet Local Area Network, the DMS-15's hard disk handles the network's central files, applications programs, electronic mail, and all network management. Instantly. Automatically.

HiNet is a complete hardware/software network—so inexpensive that you can add other work stations anywhere your business requires for about the cost of a good typewriter. Because HiNet is a network of interconnected computers, you can get information from other departments instantly.

The CP/M®-based DMS-15 combines a 64K Z80A processor, 15MB hard disk (plus 614K floppy storage), three RS-232C serial ports, the HiNet networking port, and 12 (x3) programmable function keys. Bit-mapped graphics display, too.

Get your network starter kit now. The DMS-15.

Digital Microsystems ™

Because man was not meant to work alone.

1755 Embarcadero, Oakland, CA 94606 (415) 532-3686, TWX 910-366-7310
Tavistock Industrial Estate, Ruscombe, Twyford, Berkshire, U.K., Tel. 0734-343885, Telex 849925

HiNet is a trademark of Digital Microsystems.
CP/M® is a registered trademark of Digital Research, Inc.

Circle 147 on Inquiry card.

Late Developments

Ron Rivest, one of the authors of the RSA public key cryptosystem, reports that it is presently finding commercial application in the transmission of keys for the U. S. Data Encryption Standard, a conventional system that can process information at a much faster rate. He and the other authors of the system are now at work producing a single-chip implementation of the system that can be used on a microprocessor bus, which should be able to process about 150 characters per second.

In a related item, Adi Shamir, another of the RSA authors, claims to have broken a rival public key system called the Knapsack System. Shamir's report, however, remains to be interpreted, and some variations of the Knapsack technique may still be usable. This system, developed by Ralph Merkle and Martin Hellman, is based on a well-known problem of determining which numbers of a given set of numbers were added together to produce a given sum.

on record. If, for example, Mary has filed a public key in some public access file, she can digitally sign a message to you by *decrypting* it with her private key before transmitting it. After receiving the message, you (or anyone else) can read the message by *encrypting* it with Mary's public encryption key. The process is essentially the reverse of the cryptosystem: the message is first decrypted and then encrypted, and anyone can reveal the message, but only Mary with her secret decryption key can create it.

In addition, messages using digital signatures can be subsequently encrypted with *another* key. After Mary decrypts her message to you with her secret decryption key, she can then *encrypt* it with your public encryption key. The result is a message that only Mary could have created, and only you can read!

Messages with digital signatures have other interesting and useful properties and may be used to advantage with other (non-PKC) cryptosystems. These properties and applications might easily justify an article on digital signatures alone.

Summary

This article has described the principles of public key cryptosystems. One example has been given, the Rivest-Shamir-Adleman system. We have seen how keys are constructed and used, and have at our disposal four BASIC programs for further experimentation. These programs may also be useful as models for assembly-language programs that could manipulate larger numbers and run faster. We have seen that the RSA cryptosystem provides public keys in more than astronomical quantities and that it is believed to be unbreakable.

Several questions come to mind: Is a personal computer powerful enough to run a full-size RSA system? How long would a small computer take to construct a 200-digit key? Or even a 100-digit key? How long would it take to decrypt a medium-length message?

Regardless of the answers to these questions, the prospects are good for using public key systems with small computers. New computer models appear almost monthly, and their performance is improving rapidly. The theoretical work that gave birth to the RSA system is also proceeding at a rapid pace, and we can expect new and different public key systems to result from that work. Some of these may be suitable, perhaps even optimized, for small machines, and the prospects are exciting.■

References

1. Diffie, W. "Privacy and Authentication: An Introduction to Cryptography." *Proceedings of the IEEE*, Vol. 67, March 1979, pages 397-427.
2. Diffie, W. and M. E. Hellman. "New Directions in Cryptography." *IEEE Transactions on Information Theory*, Vol. IT-22, No. 6, November 1976, page 644.
3. Knuth, Donald E. *The Art of Computer Programming: Semi-Numerical Algorithms, Volume 2*, 2nd ed. Reading, MA: Addison-Wesley, 1981,
4. Nyberg, Jostein. "A Fast, Ancient Method for Multiplication." BYTE, October 1981, page 376.
5. Rivest, R. L., A. Shamir, and L. Adleman. "A Method for Obtaining Digital Signatures and Public Key Cryptosystems." *Communications of the Association for Computing Machinery*, Vol. 21, No. 2, February 1978, page 120.