UNOS 1335

Bell Laboratories

Cover Sheet for Technical Memorandum

Title- **UNIX File Security**

Date- **January 19, 1979**

TM- **79-1271-3**

Other Keywords- **Privacy**
Encryption
Secret Writing

Author	Location	Extension	Charging Case- 39199
Robert Morris	**MH 2C524**	**3878**	Filing Case- **39199-11**

ABSTRACT

This paper describes the history of the design of the file encryption scheme on the UNIX time-sharing system. The present design was the result of countering observed attempts to penetrate the system. The result is a well-understood scheme that is fast and easy to use.

Pages Text 6	Other 1	Total 7	
No. Figures 0	No. Tables 0	No. Refs. 10	

SEE REVERSE SIDE FOR DISTRIBUTION LIST

BELL TELEPHONE LABORATORIES, INC.

TM-79-1271-3

DISTRIBUTION
(REFER GEI 13.9-3)

COMPLETE MEMORANDUM TO

CORRESPONDENCE FILES

OFFICIAL FILE COPY
PLUS ONE COPY FOR
EACH ADDITIONAL FILING
CASE REFERENCED

DATE FILE COPY
(FORM E-1328)

10 REFERENCE COPIES

<AHO,ALFRED V
<BALENSON,CHRISTINE M
<BARON,ROBERT V
<BECKER,RICHARD A
BROWN,W STANLEY
<CHEN,STEPHEN
<CHERRY,LORINDA L
<FELDMAN,STUART I
<FRASER,A G
<GOLDSTEIN,A JAY
<GRAHAM,R L
+HANNAY,N B
<JOHNSON,STEPHEN C
<KEESE,W M
<KERNIGHAN,BRIAN W
<LUDERER,GOTTFRIED W R
<MARANZANO,J F
<MARKY,GERALDINE A
+MC DONALD,H S
<MC GILL,R
MCILROY,M DOUGLAS
<MENNINGER,R E
<MORGAN,SAMUEL P
+PRIM,R C
<RALEIGH,T M
<RIDDLE,GUY G
<SCHLEGEL,C T
<SETHI,RAVI
<STORER,JAMES A
<SZYMANSKI,THOMAS G
TERRY,M E
<WEINBERGER,PETER J
<YAMIN,ELAINE E
<YANNAKAKIS,MIHALIS
34 NAMES

COVER SHEET ONLY TO

CORRESPONDENCE FILES

4 COPIES PLUS ONE
COPY FOR EACH FILING
CASE

AAGESEN,JOHN
ABATE,JOSEPH

COVER SHEET ONLY TO

ACAMPORA,A S
ACKERMAN,A FRANK
ACKERMAN,J T
ACKROFF,JOHN M
AHRENS,RAINER B
ALCALAY,D
ALLEN,JAMES R
ALLEN,R C
ALLEN,ROBERT B
AMITAY,N
AMOSS,JOHN J
AMRON,IRVING
ANDERSON,KATHRYN J
ANDERSON,MILTON M
ANDREWS,R J
ANTONIAK,CHARLES E
ARNOLD,GEORGE W
ARNOLD,PHYLLIS A
ARNOLD,S L
ARNOLD,THOMAS F
ARTHURS,E
ASMUTH,RICHARD L
ASTHANA,ABHAYA
ATAL,BISHNU S
BACH,MAURICE J
BAGGA,YUDHVEER S
BAGLEY,JOHN D
BAILY,DAVID E
BALL,MARSHALL
BARNETT,W T
BARNHARDT,KARL R
BAROFSKY,ALLEN
BARRESE,A L
BARR,W J
BASEIL,RICHARD J
BAUER,BARBARA T
BAUER,HELEN A
<BAUGH,C R
BAYER,D L
BEAUCHAMP,HARRY L
BEDNAR,JOSEPH A,JR
BENCO,DAVID S
BENISCH,JEAN
BENNETT,RAYMOND W
BENNETT,WILLIAM C
BERGH,A A
BERGLAND,G D
BERNSTEIN,DANIELLE R
BERNSTEIN,L
BERRYMAN,EDWARD R
BEYER,JEAN-DAVID
BHARUCHA,BEHRAM H
BIANCHI,M H
BICKFORD,NEIL B
BILLINGTON,MARJORIE J
BILOWOS,R M
BIRCHALL,R H
BIREN,IRMA R
BISHOP,J DANIEL
BISHOP,VERONICA I

COVER SHEET ONLY TO

BLAZIER,S D
BLEECKER,SAMUEL E
BLINN,J C
BLOSSER,PATRICK A
BLUE,JAMES L
BLUMER,THOMAS P
BLUM,MARION
BOBILIN,R T
BOCKUS,ROBERT J
BODEN,F J
BOEHM,KIM R
BOESE,J O
BOIVIE,RICHARD H
BONANNI,L E
BOND,HOLTON C,JR
BORISON,ELLEN A
BOSE,DEBASISH
BOSWELL,PAULA S
BOULIN,D M
BOURNE,STEPHEN R
BOWERMAN,REBECCA ELAINE
BOWYER,L RAY
BOYCE,C D
BOYCE,K J
<BOYCE,W M
BOYER,PHYLLIS J
BOYLE,GERALD C
BOYLE,W S
BOZA,L B
BRADFORD,EDWARD G
BRADLEY,M HELEN
BRAM,ALAN
<BRANDT,RICHARD B
BRAUNE,DAVID P
BREEN,R S,JR
BRENSKI,EDWIN F
BRESLER,RENEE A
BRIGGS,GLORIA A
BRITTON,DIANNE ELLEN
BROSS,JEFFREY D
BROVMAN,INNA
BROWN,A B,JR
BROWN,ELLINGTON L
BROWN,LAURENCE MC FEE
BROWN,ROBIN L
BROWN,W R
BROZ,G C
BUCZNY,F A,JR
BULFER,A F
BULLEY,R M
BURGESS,JOHN T,JR
BURG,F M
BURKE,P J
BURNETTE,W A
<BURNETT,DAVID S
BUROFF,STEVEN J
BUTLER,THOMAS W
BUTLETT,DARRELL L
BUTTON,BEVERLY
BUTZIEN,PAUL E

COVER SHEET ONLY TO

BYERLEE,R W
BYER,TREVOR C
BYRNE,EDWARD R
CABLE,GORDON G,JR
CAMPBELL,JERRY H
<CANADAY,RUDD H
CANNON,LAYNE W
CARDOZA,WAYNE M
CAREY,J H
CARLSON,HELEN V
CARROLL,J DOUGLAS
CARTER,DONALD H
CASEY,J L
CASPERS,BARBARA E
CASTELLANO,MARY ANN
CASTELLI,LEONARD P
CATO,H E
CAVINESS,JOHN D
CERMAK,I A
CHADDHA,R L
CHAFFEE,N F
CHAIKIN,JAMIE A
CHAI,D T
CHAKRIN,LEWIS M
CHAMBERS,B C
<CHAMBERS,J M
CHANG,G K
CHANG,S-J
CHAPMAN,LESLEY A
CHAYUT,IRA G
CHELLIS,ALICIA L
CHENG,Y
CHEN,E
CHEN,T L
CHESSON,GREGORY L
CHIANG,T C
CHILDS,CAROLYN
CHI,MELY CHEN
CHODROW,M M
CHRISTENSEN,C
CHRISTENSEN,KENT R
CHRISTENSEN,S W
CHRIST,C W,JR
CHU,PAUL H N
CIEMINSKI,DEBRA P
CIRILLO,C
CLARK,DAVID L
CLAYTON,D P
CLIFFORD,COURTENAY B
CLINE,LAUREL M I
CLOUTIER,J
COBEN,ROBERT M
COCHRAN,ANITA J
COCHRAN,JAMES A
COHEN,DAVID
COHEN,HARVEY
COHEN,HARVEY S
COKE,ESTHER U
COLABRARO,CHRISTINE A
<COLE,LOUIS M

COVER SHEET ONLY TO

COLE,MARILYN O
COLLICOTT,R B
COLLURA,THOMAS F
CONDON,J H
CONKLIN,DANIEL L
CONNERS,RONALD R
COOK,T J
COCMER,STEPHEN D
COOPER,ARTHUR E
COOPER,C ANTHONY
COOPER,MICHAEL H
COPP,DAVID H
COTTRELL,JENNIE L
CRAGUN,DONALD W
CRISTOFOR,EUGENE
<CRUME,L L
CRUPI,JOSEPH A
CUTLER,C CHAPIN
DAGNALL,C H,JR
DANKO,ANITA
DAVIDSON,CHARLES LEWIS
DAVID,ALEXANDER J
DAVIS,R DREW
DAYEM,ALY H
DE FAZIO,M J
DE GRAAF,D A
DE LESSIO,N X
DE TREVILLE,JOHN D
DEAN,JEFFREY S
DEBUS,W,JR
DENES,P B
DENNY,MICHAEL S
DESCLOUX,A
DESMOND,JOHN PATRICK
DI PIETRO,R S
DIB,GILBERT
DICKMAN,BERNARD N
DICKSON,MARY ANN T
DIESEL,MICHAEL E
<DIMMICK,JAMES O
DINEEN,THOMAS J
DITZEL,DAVID R
DIVAKARUNI,R S
DOEDLINE,BARBARA ANN
DOLATOWSKI,VIRGINIA M
DOLOTTA,T A
DOMANS,DAWN H
DOMBROWSKI,F J
DONNELLY,MARGARET M
DOUGHER,J P,JR
DOWDEN,DOUGLAS C
DOWDEN,IRIS S
DOYLE,JOHN N
DRAKE,LILLIAN
DRYBURGH,PAULINE F
D'ANDREA,LOUISE A
DUCHARME,ROBERT LAWRENCE
DUDLEY,ELIZABETH H
DUFFY,F P
DUGGER,DONALD D

...
1055 TOTAL

+ NAMED BY AUTHOR > CITED AS REFERENCE < REQUESTED BY READER (NAMES WITHOUT PREFIX WERE SELECTED USING THE AUTHOR'S SUBJECT OR ORGANIZATIONAL SPECIFICATION AS GIVEN BELOW)

MERCURY SPECIFICATION.....

COMPLETE MEMO TO:
127-SUP

COVER SHEET TO:
12-DIR 13-DIR 127

COCSFS = COMPUTER FILE SYSTEMS
COPLIP = COMPUTING/PROGRAMMING LANGUAGES/TEXT PROCESSING, EDITING
MAPRIT = MATHEMATICS/PROBABILITY/INFORMATION THEORY

HO CORRESPONDENCE FILES
HO 5C101

TM-79-1271-3
TOTAL PAGES 7

TO GET A COMPLETE COPY:

1. BE SURE YOUR CORRECT ADDRESS IS GIVEN ON THE OTHER SIDE.
2. FOLD THIS SHEET IN HALF WITH THIS SIDE OUT AND STAPLE.
3. CIRCLE THE ADDRESS AT RIGHT. USE NO ENVELOPE.
4. INDICATE WHETHER MICROFICHE OR PAPER IS DESIRED.

PLEASE SEND A COMPLETE

( ) MICROFICHE COPY ( ) PAPER COPY

TO THE ADDRESS SHOWN ON THE OTHER SIDE.

Bell Laboratories

Subject: **UNIX File Security**
Case- 39199 -- File- 39199-11

date: **January 19, 1979**

from: **Robert Morris**

TM: **79-1271-3**

MEMORANDUM FOR FILE

1. GENERAL

Most computers running under the UNIX time-sharing system can potentially be accessed by telephone by anyone with a computer terminal who cares to dial the number — and from anywhere in the world. Because of this potentially unlimited access to computers using the UNIX operating system, we have felt a responsibility to provide facilities for protecting our programs, our data, and our computer resources from unauthorized users.

Of course, the problem could be solved by denying or by placing restrictions on dial-up access, but we and others have strong special reasons for wanting to preserve this kind of access.

The protection that we require has at least these three targets:

a) security of system resources such as computer time and storage, and access to computer peripherals,

b) security of confidential, private, and proprietary information that is stored in the files of the computer — and here protection is desired both against outsiders and against unauthorized insiders, and

c) security of files not only as they reside in the storage of the machine but also as they are transmitted from machine to machine.

We want to prevent unauthorized access to our data and unauthorized use of our resources. We have taken some of the necessary steps to do just that.

This question of security is one of growing importance for us — growing in the literal sense. It will almost certainly become crucial in the future, but also it has up to now been relatively unimportant. For this reason, we have been able to treat computer security on UNIX as a research question rather than as a burning issue with tight deadlines. We started working on it early enough.

Login control is the prime defense of our data and our resources. All of our UNIX systems require (or can be made to require) a password before any use of the machine can be made. If someone cannot log onto our machines, they cannot, without physical access, get at the data stored on the machines or use our resources.

But login control is not enough protection — and for a number of reasons. Many kinds of information must be protected even from those who are authorized to log into a system. People who are not meant to read information stored on a machine may nevertheless have physical access to the machine. It is impossible to prevent persons who are authorized to log in from lending their passwords to others. People on occasion leave their terminals logged in and unattended.

The conclusion is that we need a method of file protection beyond what is provided by login control.

2. FILE SECURITY

We have concluded, from considerable experience, that files cannot credibly be protected merely by file access controls, i.e., merely by administrative software. The files have a physical existence on disk drives, on backup tapes, in system buffers, and they are on occasion transmitted from machine to machine. In addition, the administrative controls imposed by software can, and in our experience, often do fail.

We turned to cryptography to provide file security for those who desired it. The general aim was to provide security comparable to that of a secure, locked file cabinet. In other words, we have not attempted to provide an encryption scheme that would resist the determined attacks of a national government, but we have tried to make a system that is secure against the attack of an intelligent graduate student, or of a skilled ex-employee. We want, after all, not just good routines; we want also to be able to estimate just how good they are. This naturally drew us into the realm of cryptanalysis.

With the cleartext gone from the storage of the machine, the risk of inadvertent disclosure by errors of the administrative software or by operator errors was gone. Files could no longer be obtained by accident, but only by work. We have tried to increase this work, while making it as easy as possible for the user to encrypt his files.

We decided at the outset not to live in a fool's paradise:

1. We did not try to hide the relevant programs (rather we published them openly).
2. We did not make up very complex schemes that we really did not understand (rather we used the simplest schemes at each step that would just barely meet the objectives).
3. We did not try to discourage work on breaking the schemes (rather we encouraged attempts to break them and helped people who wanted to try).

Only in this way could we get the kind of critical attack needed to improve the algorithms, to make them more resistant to attack, to keep them as simple and as fast as possible, and to be able to estimate the time and labor necessary to break them. The early file encrypting routines were deliberately written so that they would be barely adequate — mainly in order to encourage attempts to break them. Each time our code breaking abilities became good enough to endanger the encryption scheme, the scheme has been replaced with a slightly better one.

There have been three versions of the UNIX encryption scheme over the past five years or so, each of which has withstood cryptanalytic attack for a considerable period of time. As this paper is intended to inform the reader about the important issues involved, I also include a discussion of three abortive versions which were flawed at the outset and only survived for a few hours or a few days.

The encrypting algorithm was written not only as a stand-alone program, but also as an integral part of the UNIX editor. This was done in such a way that an encrypted file is as easy to create and to edit as a cleartext file.

3. VERSION I

The first version of UNIX encryption was patterned after the World War II field cipher machine called the M-209 [Ref. 1, pp.425-434; Ref. 2; Ref. 5]. It never has been any great secret that cryptograms produced by this machine were routinely solved by hand during World War II, but little or nothing has been published in the past about how such cryptanalysis was done. We did not know how much text was required, how much labor it required, and whether access to cleartext or other breaches of cryptographic discipline were required.

Immediately after this encryption scheme was installed, work began on cryptanalysis of it. Methods were developed of recovering the internal settings of the machine given 50 to 75 characters of paired cleartext and ciphertext [Ref. 2]. Subsequently, methods were developed for

decryption from ciphertext alone which required only about 2000 characters of text (about 300 words) and did its work in about 2 minutes on a PDP-11 [Ref. 3]. This work destroyed the credibility of the M-209 for our purposes, and the credibility of any similar methods and the encryption program was, in any case, too slow.

By the time this analytic work was completed, version I had been replaced by

4. VERSION II

Version II was patterned after a very early rotor machine designed by an American named Hebern in about 1922 [Refs. 6,7,8] — that machine was a straight-through rotor machine that was the predecessor of the later, more famous rotor machines such as the German Enigma, the British TYPEX, and the American SIGABA [Ref. 1, p.411ff.].

Although a rotor in a rotor machine is an ingenious electromechanical contrivance, it merely amounts to a permutation which changes when the rotor moves to a new position. The changing permutation caused by one rotor (in the Enigma machine) can be described by

$$x = P(c+n1)-n1$$

where c is the cleartext character and x is the ciphertext character, both treated as integers in the range 0-25. $n1$ is an integer which corresponds to the position of the rotor; all computations are performed modulo 26. Thus a rotor performed a simple substitution on the 26-letter alphabet, and the simple substitution could change during the encryption of a message.

I used a two rotor version of this machine — the simplest version that had any chance of working. On the other hand, the rotors were increased in size from 26 (the size of Hebern's alphabet) to 256, to cover every possible value of an 8-bit byte. It turns out that the increase in size of the rotors from 26 to 256 makes the method a great deal more secure and greatly outweighs the reduction of the number of rotors from three to two.

The rotor wirings and starting positions for the machine were selected by asking the user for a typed key which could be up to 8 characters in length. This key was used as the seed for a pseudo-random number generator which was then used to select the permutation upon which the machine was based.

The heart of the encryption method is a permutation on 256 elements which is stored as a table with 256 entries. The key provided by the user is used as a seed for a random number generator which produces 2048 (=256*8) bits of output. These bits are used to choose a random permutation which we shall call P. The inverse permutation P^{-1} is computed from P. Once this has been done, we are ready to encrypt. A byte (called c) is read from the input and the following computation performed to produce the ciphertext letter x. c and x are treated as integers in the range 0-255.

$$x = P^{-1}(n2-P(c+n1))-n1$$

where $n1$ and $n2$ are integers and all computations are performed mod 256. After each character is encrypted, $n1$ is incremented and when $n1$ is incremented from 255, it becomes 0 and $n2$ is incremented.

This method requires two table lookups and four integer additions per encrypted character (including the incrementation of $n1$). The method can obviously also be rather simply implemented in hardware. The algorithm as it stands has the desirable property that it is easy to start encryption or decryption at any point, not necessarily at the beginning of the text, by setting $n1$ and $n2$ to the appropriate values. The algorithm is reflexive so that decryption and encryption become identical processes.

The numbers $n1$ and $n2$ can either be initialized to zero or to a number determined from the key. The increment at each step need not be 1 but can be any odd number (chosen, for example, from the key). In fact, the numbers $n1$ and $n2$ were initialized to zero and the increment at each step was in fact 1.

The user could type a key up to eight characters in length, and so there were about $95^8 \approx 10^{16}$ typed keys available to the user. In addition, the total number of different rotor wirings, each of which produced a different encryption, was equal to $256! \approx 10^{507}$. Both of these numbers are astronomical when one considers trying to find the key by external search. But the first installed version of this scheme used a pseudo-random number generator which was started from a 16-bit seed selected from the typed key. Therefore, although the number of user keys was astronomically large, the number of internal settings that were actually selected was only 2^{16}. It became possible to simply try each of these $2^{16} = 65{,}536$ internal settings on a cryptogram, then decrypt a few characters of it to see if it made any sense, and if not, to try the next one. Since it took only a few milliseconds to try each internal setting, the whole process of trying every possible internal setting against a cryptogram took only about two minutes. This was discovered by Dennis Ritchie acting as *avocatus diaboli*, only a matter of hours after the program was installed and prompted the move to a 64-bit pseudo-random number generator. With this change, the scheme survived for many months before any successful attack was made on it.

Given that the number of possible internal settings that were actually used was $2^{64} = 10^{19}$, it became impractical to try them all and much more profitable to try short alphabetic keys. For example, there are only 17576 keys consisting of three alphabetics, and one could try them all. This process was made as expensive as possible by using a slow algorithm to convert from the typed key to the internal 64-bit key. In particular, this conversion was done by making 25 trips through the algorithm of the Data Encryption Standard (the DES). The typed key was used as the key for the DES and a constant string was used as the cleartext for the DES. The output of each stage was used as the input of the next stage, using the same key each time.

This process takes over one-half second and it seems unlikely that it could be speeded up by more than a factor of ten. Some of the internal tables of the DES algorithm were scrambled just to make it impossible to use any of the commercial chips. Thus it takes about two hours to try every three-letter alphabetic key and this time could, by careful coding of the DES algorithm, be speeded up to about 15 minutes. Users, however, are advised to use six-letter keys which are not names or dictionary words, and preferably to use characters which are not just letters, so as to draw from a larger alphabet. The amount of time required to try all six-letter alphabetic keys (after speeding up) would be about six months, and, if both letters and digits were used, it would be many years. Some additional discussion of this question of the choice of keys is contained in Ref. 4.

Another strength of the use of a one-way cipher such as the DES is that even if the internal rotor wirings are reconstructed, no information is obtained about the actual key typed by the user, so that if the user is in the habit of choosing only names of streets in Brooklyn (for instance) as keys, this information does not become available.

5. CRYPTANALYSIS OF VERSION II

Over the course of the next six months, considerable progress was made in cryptanalysis of this Hebern scheme by direct attack on the encrypting algorithm, rather than by exhaustive key search.

The first part of the work was the development of methods to recreate the rotor wirings and starting positions and thus the internal 64-bit key, given about 1000 characters of paired ciphertext and cleartext. In other words, if the cleartext version of any 1000-character segment of a document was available, the the whole document could also be read, and any other documents encrypted with the same key. On the other hand, this method did not recover the typed key, which was still secure.

Further work resulted in a decryption program which required about 15,000 characters of text and about 5 minutes of computer time on a PDP-11. This program did direct decryption using ciphertext alone. It required no cleartext nor any probable words. This work is reported in a forthcoming memo by Peter Weinberger.

Since this program was able to solve cryptograms well within the range of the size of actual files that might be encrypted, the algorithm had to be improved or replaced. The general method, however, seemed promising, and it was replaced by an only slightly more sophisticated scheme.

6. VERSION III

Version III of file encryption is based on the German Enigma machine of WWII fame [Ref. 1, p.420ff.; Refs. 9,10]. This version used only one rotor and a reflector, rather than the three rotors used in the Enigma.

The algorithm begins by selecting at random a permutation P on 256 letters and, in addition, another permutation R of order two (a transposition). The inverse permutation P^{-1} is computed from P. As before, a byte (called c) is read from the input and the following computation performed with c and x treated as integers in the range 0-255.

$$x = P^{-1}(R(P(c+n1)+n2)-n2)-n1$$

where $n1$ and $n2$ are integers, and all computations are performed mod 256. After each character is encrypted, $n1$ is incremented, and when $n1$ is incremented from 255, it becomes 0 and $n2$ is incremented. Again, the process is its own inverse — encryption and decryption are identical.

This method requires three table lookups and five integer additions per encrypted character. It is, like its predecessor, easily implemented in hardware. It shares the desirable property that it is easy to start encryption or decryption anywhere in the text by starting with suitable values of $n1$ or $n2$.

The number of possible initial states of the machine has been greatly increased by the addition of the reflector R. In the algorithm, only transpositions R are used which are the product of 2-cycles, and there are approximately 10^{252} of them. The result is about 10^{760} possible internal states. (Still only 2^{64} of these are selected by the pseudo-random number generator.)

7. FLAWED VERSIONS

The final (i.e. current) version III was preceded by two flawed versions of very similar algorithms. A description of these flawed versions might be instructive.

In the first attempt to improve version II I avoided having to compute 256 additional pseudo-random numbers to create the required transposition. The transposition was obtained from the permutation P, by what was thought to be a suitable transformation, to produce a new permutation of order 2 to be used for the transposition R. By a marvel of group theory, the product of the three transformations $P^{-1}RP$ turned out to be independent of the key and so whenever both $n1$ and $n2$ were zero, a block of 256 characters of cleartext appeared in the output no matter what key was used for decryption. Although this did not compromise either the key or the rest of the message, it was considered a serious flaw.

This flawed version was replaced by another which again attempted to avoid having to compute the 256 additional pseudo-random numbers to create the transposition R. Just one more pseudo-random number was computed and the transposition R consisted of forming the exclusive OR with this random byte.

This version had two serious flaws. The first was that on the average, one out of 256 ($= 2^8$) keys would produce zero as the pseudo-random byte and thus the algorithm would do no encryption whatever; its output would be the cleartext. Another flaw was that for each of the 256 possible random bytes, the algorithm was very similar to Version II and so the work factor for cryptanalysis had only been multiplied by 256, which was far too small a factor.

No further attempts were made to avoid separate computation of the additional pseudo-random numbers required to determine R.

8. WORK FACTORS

The algorithm of Version III is still theoretically subject to direct cryptanalytic attack by a combination of statistical and algebraic methods. This line of attack is well understood and our experience with similar systems leads us to believe that the amount of text required is beyond the size of files likely to be encrypted. We also believe that the machine time and storage required would be very large.

The most likely route of attack is some form of exhaustive key search, and this can take three forms:

- searching for the key typed by the user,
- searching for the 64-bit seed which drives the random number generator, and
- searching for the correct permutations P and R.

Possible permutations P and R can be tried extremely rapidly and with the right hardware, a new one could be tested every few microseconds. On a general purpose computer, perhaps as much as 50 microseconds would be required. On the other hand, there are 10^{760} of them to try, and this is too many.

It is a somewhat longer task to try one of the 64-bit quantities which drives the random number generator, to construct P and R, and to test them against the encrypted text. Perhaps one could be tested per millisecond. There are 10^{19} of them, and this is also too many.

Slowest of all is the task of trying a promising typed key. With a suitable rewritten program for the DES, a new key could be transformed into a permutation and tested in less than 50 milliseconds. This is by far the most promising approach, since the whole set of three-letter alphabetic keys, or every word in a small dictionary could be tried in 15 minutes or so. This matter of choice of keys is extremely important and it is a problem shared by all key-based encryption algorithms. It is for this reason that we urge all users to use longer and more complex keys. In particular, we urge the use of keys of at least six characters in length which form neither a word nor a name.

9. CONCLUSIONS

By now, we have had considerable experience in the varieties of attacks on cryptographic systems and we have more information on which to base an estimate of the strength of the current system — how much time and labor it would require, and how much text might be required for successful cryptanalysis. Although the current scheme has no known weaknesses, no doubt we will continue to improve our techniques as time goes on and to install new schemes when it proves necessary.

It is worthwhile to point out here that encryption can form only a part of a system that ensures the security of data. If the adversary has unsupervised physical access to the computer, to key materials, or to the cleartext versions of encrypted files, then he cannot, in the long run, be prevented from getting at the data. Nor can the system administrators (and perhaps the author of the encryption programs) be prevented from getting access to the data. A system administrator could, for example, install his own version of the encryption programs to do whatever he likes — for example mailing all of the keys to himself.

The whole matter of selection, custody, and distribution of keys deserves close attention from anyone who cares about the security of his data. If a user chooses his or her spouse's name as a key for his encrypted files, then nothing the operating system can do will help him.

MH-1271-RHM-unix

Robert Morris

Att. 1
References

References

[1] David Kahn. *The Codebreakers.* MacMillan, New York (1967).

[2] Robert Morris. The Hagelin Cipher Machine (M-209) — Reconstruction of the Internal Settings. **TM78-1271-4,** March 23, 1978.

[3] James Reeds, Dennis Ritchie and Robert Morris. The Hagelin Cipher Machine (M-209) — Cryptanalysis from Ciphertext Alone. **TM78-1273-2, July 24, 1978.**

[4] Robert Morris and Ken Thompson. UNIX Password Security. **TM78-1271-5,** April 3, 1978.

[5] U.S. Patent #2,089,603

[6] U.S. Patent #1,683,072

[7] Cryptologia, Vol. I, No. 2 (April 1977) pp.168-185.

[8] Popular Mechanics, December 1922, pp.849-850.

[9] U.S. Patent #1,657,411

[10] Gebrauchsanleitung für die Chiffriermaschine Enigma, **H. Dv. g. 13,** Reichsdruckerei, Berlin, 1940.