UNOS 1345

Bell Laboratories Cover Sheet for Technical Memorandum

The information contained herein is for the use of employees of Bell Laboratories and is not for publication. (See GEI 13.9-3)

Title- **An Essay in Computer Security: Decrypting A Former UNIX† *crypt***

Date- **December 29, 1978**

TM- **78-1271-20**

Other Keywords- **Ciphers**

Author	Location	Extension	Charging Case- 39199
P.J. Weinberger	**MH2C-514**	**7214**	Filing Case- **39199-11**

ABSTRACT

Users of computer systems need some way of keeping their files private from non-privileged users sharing the same machine. On UNIX at least, relying on the file protection mechanism decreases the amount of legitimate sharing which is possible, and leaves the file readable on dump tapes. A series of encryption schemes have been tried on the Center 127 system. The next-to-last of these, used from March to December of 1978, seems to be quite satisfactory for small files However, the techniques discussed in this memo generally recover at least 70% of the plain text of the file given about 11000 characters of enciphered text. The cryptanalysis algorithm takes about two minutes of processor time. Another 10 minutes of human intervention usually completely decrypts the text.

†UNIX is a Trademark of Bell Laboratories.

TM-78-1271-20

Pages Text 5	Other 0	Total 5
No. Figures 0	No. Tables 0	No. Refs. 0

BIREN,IRMA B
MH2F128
SUBJECT MATCH

CM
01/31/79
COCSFS

E-1932-U (6-73) SEE REVERSE SIDE FOR DISTRIBUTION LIST

DISTRIBUTION
(REPER 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
ANDREWS,E J
ARTHURS,E
<CALENSON,CHRISTINE M
BARNHARDT,KARL R
<CARON,ROBERT V
BAYER,D L
<BECKER,RICHARD A
BHARUCHA,BEEBAM H
BIREN,IRMA B
BLUM,MARION
BOBILIN,R T
BOWYER,L RAY
BOYCE,K J
BOYCE,W M
BOZA,L B
BROWN,ELLINGTON L
BROWN,ROBIN L
BROWN,W R
BROWN,W STANLEY
BURKE,P J
BUTLETT,DARRELL L
CARTER,DONALD H
CERMAK,I A
CHADDHA,R L
CHAI,D T
CHANG,S-J
<CHEN,STEPHEN
CHEN,T L
<CHERRY,LORINDA L
CHIANG,T C
COHEN,DAVID
DE FAZIO,M J
DE GRAAF,D A
DE LESSIO,N X
DESCLOUX,A
DIB,GILBERT
DICKMAN,BERNARD N
DIVAKARUNI,R S
DONNELLY,MARGARET M
ELY,T C
FABRICIUS,WAYNE N
<FELDMAN,STUART I
FLEISLEBER,RAYMOND C
FORYS,L J
FOUGHT,B T
FOUNTOUKIDIS,A
FRANK,AMALIE J
<FRASER,A G
FREEMAN,K G
FROST,H BONNELL
FU,C
GABEY,MICHAEL R
GEORGEN,MICHAEL R
GEYLING,F T
GILBERT,E N
GITLIN,R D
GOLDBERG,HAROLD JEFFREY
GOLDMAN,JOEL
<GOLDSTEIN,A JAY
GORDON,MOSHE B
<GRAHAM,R L
GUIDI,PIER V
HACHENBURG,V
HALL,H M,JR
HALL,WILLIAM G
HANDLER,GARY J
+HANNAY,N B
HANSEN,GWEN J
HARTMANN,E H
HERMAN,KENNETH M
+HOLCOMB,J E
HOLTMAN,JAMES P
HOLTZMAN,JACK M
HOU,TIEN-FANG
HO,TIEN-LIN
HSU,TAU
HYDE,THOMAS ALLEN
IPPOLITI,O D
IRVINE,M M
IVIE,EVAN L
JACOBS,IRA
JAGERMAN,D
JAIN,ARIDAMAN K
<JOHNSON,STEPHEN C
KANE,JOHN M
KANODIA,RAJENDRA K
KAPLAN,FRANK
<KEESE,W M
KELLY,L J
<KERNIGHAN,BRIAN W
KOETHER,MARY E
LANDOLINA,W C,JR
LEVENBERG,R A
LEWINE,R N
LEWINSKI,D A
LEWIS,DALE ELIZABETH
LIEN,Y EDMUND
LIM,Y S
LORENC,A
LUCKY,R W
<LUDERER,GOTTFRIED W R
MAHLER,GLENN R
MALLOWS,COLIN L
<MARANZANO,J F
<MARKY,GERALDINE A
MARLOW,NORMAN A
+MC DONALD,H S
<MC GILL,R
MC GONEGAL,CAROL A
MC NEES,GARY K
MC SKIMIN,J R
MCILROY,M DOUGLAS
MEADORS,HOWARD C,JR
<MENNINGER,R E
<MORGAN,SAMUEL P
+MORRIS,ROBERT
MORRIS,ROBERT J T
NIMTZ,ROBERT O
O SHEA,W T
ODLYZKO,A M
ODONOHUE,J P
OEHRING,H
OPITZ,R J
PACK,CHARLES D
PEARLMAN,MARY ELLEN
PENNISI,CONCETTO J
PHILLIPS,BRIAN J
PINSON,ELLIOT N
POLLAK,H O
+PRIM,R C
<RALEIGH,T M
REES,W M
<RIDDLE,GUY G
ROBERTS,CHARLES S
RODRIGUEZ,ERNESTO J
ROSLER,LAWRENCE
ROY,SUMIT
RUBIN,PHILIP E
RUSSO,JACK
SAPERSTEIN,B
SATZ,LAWRENCE R
<SCHLEGEL,C T
SCHONFELD,TIBOR J
SEAGREN,M K
<SETHI,RAVI
SEYMOUR,ROBERT F
SHERMAN,R
SHIVELY,R R
SHOENFELT,J E
SHORTER,J W
SINGH,R P
SKUBIAK,MICHAEL T
SLOANE,N J A
SPIRES,R J
STETTNER,ARMANDO P
<STOBER,JAMES A
STROHECKER,CARY A
STROM,B IVAN
STURMAN,JOEL N
SWAMINATHAN,K R
SWANN,DALE W
SWAYNOS,JOHN J
<SZYMANSKI,THOMAS G
TERRY,M E
THOMPSON,J S
THOMPSON,K
THOMPSON,RICHARD A
WANG,Da-LUN
WATSON,L S
<WEINBERGER,PETER J
WILLIAMS,W H
WILSON,D E
WITSENHAUSEN,H S
WUNDERLICH,EBERHARD F
WYNER,AARON D
<YAMIN,ELAINE E
<YANNAKAKIS,MIHALIS
179 NAMES

COVER SHEET ONLY TO

CORRESPONDENCE FILES

4 COPIES PLUS ONE
COPY FOR EACH FILING
CASE

ABATE,JOSEPH
ACAMPORA,A S
ACKERMAN,A FRANK
ACKERMAN,J T
AHRENS,RAINER B
ALCALAY,D
ALLEN,JAMES R
AMITAY,N
AMOSS,JOHN J
AMRON,IRVING
ANDERSON,KATHRYN J
ANDERSON,MILTON M
ANTONIAK,CHARLES E
ARNOLD,GEORGE W
ARNOLD,S L
ARNOLD,THOMAS F
ASMUTH,RICHARD L
ASTHANA,ABHAYA
ATAL,BISHNU S
BACH,MAURICE J
BAGLEY,JOHN D
BALL,MARSHALL
BARNETT,W T
BAROFSKY,ALLEN
BARRESE,A L
BASEIL,RICHARD J
BAUER,BARBARA T
<BAUGH,C R
BEDNAR,JOSEPH A,JR
BENISCH,JEAN
BENNETT,RAYMOND W
BENNETT,WILLIAM C
BERGLAND,G D
BERNSTEIN,DANIELLE R
BERNSTEIN,L
BERRYMAN,EDWARD R
BEYER,JEAN-DAVID
BICKFORD,NEIL B
BILOWOS,R M
BIRCHALL,R H
BISHOP,J DANIEL
BLAZIER,S D

COVER SHEET ONLY TO

BLEECKER,SAMUEL E
BLINN,J C
BLOSSER,PATRICK A
BLUE,JAMES L
BOCKUS,ROBERT J
BODEN,F J
BOESE,J O
BOIVIE,RICHARD H
BOSE,DEBASISH
BOURNE,STEPHEN R
BOYCE,C D
BOYER,PHYLLIS J
BOYLE,W S
BRADLEY,M HELEN
BRAM,ALAN
<BRANDT,RICHARD B
BRENSKI,EDWIN F
BRITTON,DIANNE ELLEN
BROSS,JEFFREY D
BROVMAN,INNA
BROWN,A B,JR
BROWN,LAURENCE MC FEE
BROZ,G C
BUCZNY,F A,JR
BULFER,A F
BURGESS,JOHN T,JR
BURG,F M
BURNETTE,W A
<BURNETT,DAVID S
BUROFF,STEVEN J
BUTTON,BEVERLY
BYERLEE,R W
BYER,TREVOR C
BYRNE,EDWARD R
CAMPBELL,JERRY H
<CANADAY,RUDD H
CANNON,LAYNE W
CARDOZA,WAYNE M
CAREY,J H
CARROLL,J DOUGLAS
CASEY,J L
CASPERS,BARBARA E
CASTELLANO,MARY ANN
CASTELLI,LEONARD P
CATO,H E
CAVINESS,JOHN D
CHAFFEE,M P
CHAIKIN,JAMIE A
CHAMBERS,B C
<CHAMBERS,J M
CHANG,G K
CHAPMAN,LESLEY A
CHAYUT,IRA G
CHENG,Y
CHESSON,GREGORY L
CHILDS,CAROLYN
CHODROW,M M
CHRISTENSEN,C
CHRISTENSEN,KENT R
CHRISTENSEN,S W
CHRIST,C W,JR

...
710 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

COCSFS = COMPUTER FILE SYSTEMS
MAPBIT = MATHEMATICS/PROBABILITY/INFORMATION THEORY

COVER SHEET TO:
12-DIR 13-DIR 127

HO CORRESPONDENCE FILES TM-78-1271-20
HO 5C101 TOTAL PAGES 5

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: **An Essay in Computer Security: Decrypting A Former UNIX† *crypt***
Case- 39199 -- File- 39199-11

date: **December 29, 1978**

from: **P.J. Weinberger**

TM: **78-1271-20**

MEMORANDUM FOR FILE

1. The former encryption algorithm

The history and design goals of the various UNIX encryption schemes are set out in some detail in a forthcoming memo by Robert Morris. (Robert Morris, UNIX File Security, TM79-1271-3) The encryption method described in this memo was used until December 1978, but, having been shown to be insecure, it has been replaced on our machine.

The old encryption scheme is driven by a key provided by the user. This key is converted into a 12 byte intermediate key using the UNIX password encryption algorithm. This step is probably irreversible without the key, as the UNIX password encryption algorithm seems to have the same degree of security as the national Data Encryption Standard. Further, the UNIX password encryption algorithm takes several hundred milliseconds, making any sizable key search prohibitively lengthy. (This is not to say that there aren't efficient ways of finding passwords; but none are known.)

The 12 byte intermediate key is converted to a pseudo-random permutation P of the 256 8-bit bytes. The cryptanalysis attempts to determine this permutation P. The number of these permutations is 256!, which is larger than 10^{505}, while there are no more than 10^{29} 12 byte intermediate keys. Nonetheless, no way is known to take advantage of the fact that most permutations cannot occur.

The permutation P is used to encrypt the file as follows. Let the i-th byte of the file to be encrypted be denoted by c_i. Then the i-th byte of the enciphered file, e_i, is given by

$$e_i = -i+P^{-1}(\lfloor i/256 \rfloor - P(c_i+i))$$

where all the additions and subtractions are done modulo 256. This equation may be rearranged to give

$$P(c_i+i)+P(e_i+i) = \lfloor i/256 \rfloor$$

from which it is seen that the algorithm is self inverting. The equation also shows that there are two permutations which give the same result, namely P and $P+128$.

The encryption algorithm, while not exactly a classical rotor machine, is very similar to one. The text feeds through a rotor P, which steps one for each character, and then feeds through a reflector $i/256-x$ which changes after P has completed an entire revolution, and then feeds back through P^{-1}. In contrast to more familiar rotor machines, the alphabet is 256 characters instead of 26. Also, the rotor depends entirely on the key, while in the machines used in World War II the rotors were chosen from a fixed set (about 10) of rotors. On the other hand, the fact that the reflector changes at known character positions surely makes the

†UNIX is a Trademark of Bell Laboratories.

system less secure, and the fact that fewer than half of the characters normally occur in plain text means that each character of enciphered text should provide at least a bit of information towards reconstructing P. Further, certain regularities in the plain text are reflected by regularities in the cipher text. For instance, if

$$c_i - c_j = j - i,$$

and the reflector has not changed between c_i and c_j, then

$$e_i - e_j = j - i$$

also. The commonest cases of this are 'ed' in text files, and "newline tab" in C programs. However, it is not clear how to exploit all this information directly.

The remainder of the memo is in two parts. The first part describes reconstructing P when both plain text and enciphered text are given. There is not much to this, except for programming tricks, but the number of characters required indicates something about a lower bound for the security of the system. The second part describes reconstructing P, and thereby recovering the plain text, given a sufficient quantity of enciphered text.

2. Reconstructing P from known plain text

Reconstructing P from known plain text amounts to solving the equations

$$P(e_i+i)+P(c_i+i) = \lfloor i/256 \rfloor$$

for the 256 unknowns $P(n)$. The equations must be solved modulo 256, and, as observed above, the solution is not unique. The only obstacles to straightforward solution of these equations are time and space. On the PDP 11/70, space is the critical problem, as 256 equations in 256 unknowns will not fit in the address space of a program. Even if they did, solving that many equations by the usual triangularization algorithms takes a fair amount of time. Fortunately, it is possible to preserve the sparseness of the system while solving it, so that only a few seconds of computer time are required. An additional problem is that not enough plain text may be given, so that out of the values of the $P(n)$ consistent with the equations must be selected a set which is also a permutation. In general this seems to require an exponential search, but in cases where P is nearly determined, the search can be ordered in a way which runs quickly.

Files chosen at random, and encrypted with random keys, seem to require about 750 to 850 characters of plain text to reconstruct P. A file consisting of all new-lines, encrypted with random keys, requires only about 500 characters of plain text. Thus the equations are significantly redundant.

The equations to be solved have one of two forms. If e_i+i is different from c_i+i, then the equation has the form

$$P(a)+P(b) = c,$$

while if they are equal, the equation has the form

$$2P(a) = c.$$

If two equations of the first sort share a variable then the result of eliminating that variable from the two equations is of the form

$$P(a)-P(b) = c.$$

If any two of these three forms of equations have a variable in common, then the result of eliminating that variable from the two equations is again an equation of one of the three kinds. Therefore a straightforward elimination algorithm which only expects equations of these forms will suffice for solving the equations, and will run quite quickly.

For instance, one complete solution ended up with the following equations:

$$P(0)-P(255) = 8$$

$$P(1)+P(255) = 52$$
$$P(2)+P(255) = 210$$
$$\cdots$$
$$P(254)+P(255) = 26$$
$$2P(255) = 160.$$

Choosing one of the two possible values for $P(255)$ determines the permutation P.

Using 450 characters instead of 500 characters in the above example leads to one equation $2P(237)=0$, one equation $2P(255)=160$, 8 equations of the form $P(a)\pm P(237)=c$, so that chosing one of the possible values of $P(237)$ would determine the values of 8 other variables, 148 equations of the form $P(a)\pm P(255)=c$, so that determining a value of $P(255)$ would determine the values of 148 other variables, 25 equations of the form $P(a)\pm P(246)=c$, 31 equations of the form $P(a)\pm P(248)=c$, and 39 equations of the from $P(a)\pm P(252)=c$. The search part of the solver tries values for $P(255)$ and $P(237)$ first, since there are only two possible values for each of them, and for each choice of their values which is consistent with P being a permutation, it tries values for the remaining free variables, $P(252)$, $P(248)$, and $P(246)$, in that order, in an attempt to generate a legal P. The free variables are tried in the order which determines the most values earlier in the search. The program stops when it finds the first permutation.

3. Reconstructing P from cipher text alone

The basic equation

$$P(e_i+i)+P(c_i+i) = \lfloor i/256 \rfloor \tag{1}$$

is again the starting point, but none of the c_i are known. If it is known that

$$c_i = c_{i+256k}$$

then the equations for i and $i+256k$ can be subtracted to give

$$P(e_{i+256k})-P(e_i) = k. \tag{2}$$

These equations are of the form considered in the last section, so solving them is no problem. The difficulty is generating them.

The key to finding such equations is that the assumption that $c_{i+256k}=c_i$ is true rather often. Let ρ_c denote the fraction of the time that character c occurs in the plain text. Then if the characters that occur in the plain text separated by multiples of 256 are uncorrelated, which seems to be true, the probability that the assumption is true is

$$\gamma = \sum_c \rho_c^2.$$

For text files this is about 6%, while for C programs it is about 3.5%. For any j which is not zero, the equation

$$c_i - c_{i+256k} = j$$

leads to an incorrect version of (2). The incorrect ones appear to occur more or less uniformly. Now for each value of k and each a, there is exactly one true equation

$$P(a)-P(x) = k$$

so that for each k there are exactly 256 true equations, and 65280 false ones. Therefore, if one fixes k and makes the assumption that $c_i=c_{i+256k}$ for *all* i, good equations are 255γ times as likely to occur as false equations. Making this observation, and believing in its validity, are all that is required for the cryptanalysis.

The deciphering uses a set of programs. The first makes the above assumption for all positive k and all i. Each equation that occurs at least 4 times (4 is a constant that seems to give good behavior) is written to the output, together with the number of times it occurred. These equations are sorted so that the equations which occur most often come first. The sorted output is read by an equation solver of the form described in the last section. If an inconsistency is detected in one of the equations which occurred j times, the equation solver produces the result of using only those equations which occur at least $j+1$ times. (As will be seen below, it could still be using incorrect equations.) If there was sufficient enciphered text, the equation solver will essentially produce P. About 20000 characters seems to be enough for this.

In any case the solution of a set of these equations cannot be unique, for if P is a solution, so is $P+i$, for any i. Therefore the output of the equation solver has at least one free variable, and maybe more. This output is processed by a program which attempts to find good values for the free variables The two criteria by which the goodness of a set of values for the free variables are judged are that the resulting P should be a permutation, and given that, that decrypting the file using the trial P should give ASCII output. The program uses the amount of ASCII in the first 1000 characters of the decrypted file as a measure of goodness, and it does the same kind of tree search that is described in the last section. Each time this program finds a better P it writes it into a file, and prints the percent of ASCII on its output. On 15000 character enciphered files there are usually sufficiently few possibilities that the program can try them all in a few seconds, but sometimes it will run for a long time (hours). In this case it can be killed at any time, and the best solution found so far will be in its output file. If everything has worked well, at this point decrypting the file with the given P will result in about 70% of the characters being correct. This is enough to just read off the clear text, if the file is just text. There is also an editor-like program which takes the enciphered text and the trial P, and allows the user to correct its decipherment. It takes the corrections and calculates their effect on P and presents new pieces of deciphered text. About 10 minutes of interaction generally recovers all of P.

The reader's understanding of this process may be enhanced by describing the deciphering of an example file. The enciphered file was 13575 characters long. Therefore the equation generator considered 353139 equations, of which it turns out that 2080 occurred at least 4 times. Generating and sorting these required just over 2 minutes of cpu time. The following table shows how many equations were repeated. For instance, the first line means that 4 of the equations occurred 10 times.

10	4
9	16
8	27
7	106
6	232
5	568
4	1127

The equation solving program found its first inconsistency in an equation that occurred 4 times. Using just the equations that occurred at least 5 times left 3 free variables. This took about 3 seconds of cpu time. The next program tried to choose values of the 3 free variables but it could find no values which were consistent with P being a permutation. Therefore at least one of the equations which occurred 5 or more times must have been incorrect. At this point the equation solver was reinvoked (manually) using only equations which occurred at least 6 times. It produced a solution with 32 free variables, 28 of which occurred in no equation. That is, the solver produced 224 equations which depended on 4 free variables. The next program took 11 seconds of cpu time to find values of the 4 free variables which produced a trial P. Encrypting using the trial P produced 786 ASCII characters from the first 1000 characters of the file. At this point deciphering the file produced

As we?have s_en, this _rinc?_le ma_ _e viewed as
th?_ethi_s of _ single rati??a?_i?divi?ua?
?repar__ to t?ke_whatever chance? ?eceśs_r_ to ma_imi_e
his p_ospec?? fr?m??he?suan?qp?_t o? thf _nitial situation.
(?f ?here i_ no_obj?ctive bas?s ?or?probabili_ies, the?

The underscores and question marks indicate undecipherable characters. Most of this small sample is easily readable, but it is clear that the fourth line has some incorrect characters in it. About 10 minutes of interaction with the editor program produced the complete P. The lines corresponding to those above are actually

As we have seen, this principle may be viewed as
the ethics of a single rational individual
prepared to take whatever chances necessary to maximize
his prospects from the standpoint of the initial situation.
(If there is no objective basis for probabilities, they

It is interesting to ask how close to correct the intermediate steps were. Of the 2080 original equations, 191 of the equations which occurred 4 times were incorrect, and exactly one of the equations which occurred 5 times was incorrect. Therefore all of the equations put out by the equation solver were correct. The program that generated the trial P produced 222 correct values, 6 incorrect values, and 28 variables which it didn't try to determine. What happened was that the values of two of the free variables were guessed incorrectly.

4. Problems and Projects

Below are given several simple modifications of the basic enciphering equation. In them the parameters n_0, n_1, and so forth, are determined by the user's key. The user is invited to decide, by thought or by program, how much more complicated using one of the modifications would have made the cryptanalysis.

$$P(e_i+i+n_0)+P(c_i+i+n_0) = \lfloor i/256 \rfloor. \quad \text{a.}$$

$$P(e_i+n_0 i)+P(c_i+n_0 i) = \lfloor i/256 \rfloor. \quad \text{b.}$$

$$P(e_i+i)+P(c_i+i) = \lfloor (i+n_0)/256 \rfloor. \quad \text{c.}$$

$$P(e_i+i)+P(c_i+i) = \lfloor n_0 i/256 \rfloor. \quad \text{d.}$$

$$P(e_i+n_0 i+n_1)+P(c_i+n_0 i+n_1) = \lfloor (n_2+n_3 i)/256 \rfloor. \quad \text{e.}$$

(At least one of them makes no difference at all.)

P J Weinberger

MH-1271-PJW-unix

P.J. Weinberger