EXECUTION TIME REQUIREMENTS FOR PROGRAMMED ENCRYPTION METHODS

by

Theodore D. Friedman and Lance J. Hoffman

Memorandum No. ERL-M378

1 June 1973

ELECTRONICS RESEARCH LABORATORY

College of Engineering
University of California, Berkeley
94720

EXECUTION TIME REQUIREMENTS FOR PROGRAMMED ENCRYPTION METHODS

by

Theodore D. Friedman and Lance J. Hoffman

Department of Electrical Engineering and Computer Sciences
and the Electronics Research Laboratory
University of California, Berkeley, California 94720

ABSTRACT

Although encryption has often been discussed as a means to protect computer data, its costs are not well established. Five experiments were conducted to measure the CPU time on a CDC 6400 required by additive encryption methods programmed both in assembly language and in FORTRAN: a "null transformation" to measure the time to move data without encryption; encryption with a one-word key; encryption with a 125 word key; double key encryption; and encryption using a pseudo-random key. The results were analyzed for consistency over 100 runs, and the effects of constant and intermittent errors were considered.

Timing rates for assembly language encryption ranged from 498,800 characters per second for pseudo-random key encryption to 2,092,000 characters per second for a one-word key encryption. The latter is almost equivalent to the rate required simply to move data without encryption. FORTRAN tests required two to four times as much CPU time. This paper introduces the idea of encryption time coefficient, the ratio of encryption time to the time taken to move data without encryption.

KEY WORDS AND PHRASES: Encryption, security, privacy transformations, protection, cryptography, cryptology.

CR CATEGORIES: 2.12, 2.43, 3.70, 3.81, 4.32, 4.39

Research sponsored by National Science Foundation Grant GJ-36475.

Introduction

Encryption methods have come to be recognized as a means of protecting sensitive computer data from disclosure during transmission to remote locations or when stored on external file devices [1,2,3,4,5,6,7,8,9,10]. However the costs in computer time of encrypting data are not well established. If special purpose equipment such as shift register key generators [3,11] are used, it may be possible to encrypt data "on the fly" during input/output operations, thereby eliminating all time overhead for the central processor. But in the absence of such equipment, the transformations must be accomplished by the central processor, so that a cost penalty is imposed.

The purpose of the study described here was to investigate the cost in central processor execution time for encryption processes conducted by certain programmed algorithms.

Data may be encrypted by a number of methods, such as substitution alphabets, additive systems, transpositions, and "non-linear" [12] transformations. Of these, additive systems are undoubtedly the most suitable for implementation by program when considered in terms of convenience, speed of operation and efficient use of the central processor resources. The additive transformations involve residue addition of the digits of a key or set of keys to corresponding digits of the cleartext (untransformed) data.

Additive systems may be distinguished according to the type of key used. A one-word key is most convenient, but text encrypted by such a key is not well protected. Longer periodic keys provide stronger encryptions, i.e., the resulting cryptotext is more resistant to analysis by an unauthorized party. Double key encryption systems are similar to very long single key systems while using less storage, but the double key systems

have been shown to be considerably weaker than corresponding single key systems [13]. Finally, so-called "infinite key" systems have been proposed, in which one or more pseudo-random number generation algorithms are used to provide a continually changing key [3,4,5]. Pseudo-random systems offer very high security, but difficulties arise when data so encrypted are retrieved from random access storage. To decrypt such data, the proper section of the original pseudo-random key must be re-derived, which may be so time-consuming a process as to be impracticable.

Thus, the various additive encryption schemes have distinct advantages and shortcomings. Cost data for these processes would be valuable, but little has been published at this time. Carroll and McLelland [5] reported that they encoded 37,000 bytes of data per second using a pseudo-random key sequence on a PDP-10/50 computer. Unfortunately this is the only data they provided. Graff [14] investigated delays introduced by encryption programmed on a Control Data 6400 during access to mechanical storage media. In the case of access to disc storage, for example, he reported that 1000 80-character records were encrypted by an 80-character key in 1.07 seconds, and that the 1000 80-character records were encrypted by a pseudo-random key in 2.62 seconds. It should be noted that these results are highly specific to the mechanical media used. Conway, Maxwell, and Morgan reported that they were able to encrypt "a 500 byte record on a 370/155 in about 500 micro-seconds of CPU time" [15], but they do not describe the type of encryption except that they used "standard cryptographic techniques," nor do they supply any details about the encryption program.

Garrison and Ramamoorthy [16] investigated encryption on a Control Data 6600 computer using a variety of schemes. Using a one-word key

additive system, their report indicates that 5760 bits were encoded in .002 seconds. When a non-repreating key was used, 480 bits were encoded in .002 seconds. However, their presentation is marred by use of undefined parameters and ambiguous descriptions.

Nature of the Experiments

The present study was undertaken to provide more extensive, useful, and replicable data on the cost of encryption. The decryption process is identical to the encryption process. Therefore the cost of decryption is equal to the cost of encryption.

The CPU time required for encryption of a section of text by several different additive systems was measured. Cleartext was entered in alphabetic form from cards. Encryption consisted of addition modulo two of the binary representation of the text with the corresponding bits of the key. To avoid extraneous influences, the entire encryption process was conducted in main core memory.

The experiments for the study were programmed in assembly language and also in FORTRAN and were run on the Control Data Corporation 6400 computer at the University of California, Berkeley, under normal load conditions using the CALIDOSCOPE operating system. Five tests were made:

1. A null transformation in which the test data were simply moved from one location to another without transformation. This provided a base for comparing the execution time overhead imposed by the different encryption transformations.
2. One-word key encryption, in which a constant key, one 60-bit word long, was added modulo two ("exclusive or") to each 60 bit word of data.

3. Long key encryption, in which a periodic key 125 words long was added modulo two to each successive 125 words of data.
4. Double key encryption, in which two periodic keys, 125 and 123 words long, were added modulo two to the cleartext. The result is in some ways similar to encryption by a single key of length equal to the product of the two key lengths, namely 125x123 = 15,375 words.
5. Pseudo-random key encryption, in which a continually changing key was provided by the standard FORTRAN pseudo-random number algorithm.

It should be noted that in these tests, the addition process is actually accomplished by a single computer instruction. Differences in time requirements result from different amounts of "housekeeping" needed to produce the proper key, to prepare the data and key for addition, and to control iteration of the associated program loops. In effect, the purpose of these tests was to compare overheads imposed by the housekeeping of the different transformations.

For example, the one-word key method would be expected to take less time than the long key method because in the former case, the same key word encrypts every word of data, while in the latter case, iteration control must select a new one-word key segment for each new word of data.

Results

The objective was to determine the time required to carry out each of the encryption processes. It would have been convenient to measure the exact time required to encrypt a single word of data. However due to the lack of precision of the computer clock and because the clock readout process itself introduced a certain delay, measurements instead

were made of the time required to encrypt long consecutive segments of data. Encryption of a word of data was expected to require a few micro-seconds on the 6400, and since the readout clock is callibrated in milli-seconds, it was decided that at least 20,000 words should be processed between clock readings.

The results of the assembly language routines are considered first, and are summarized in Table I. The elapsed time was measured for processing 20,000 words of text by each of the five transformations. The average null transformation, representing the time required to execute one iteration of a loop to copy data from one location to another, was 4.76 micro-seconds. Encryption by a one-word key required an average of 4.78 micro-seconds per word. The assembly code for these two tests differed by only a single machine instruction: a register transfer in the null transformation test was replaced by a logical difference instruction. Since both instructions require the same execution time in the 6400, the time of both tests was almost identical.

The long key encryption required 8.2 micro-seconds and the double key encryption required 12.56 micro-seconds per word. The additional time in these cases is attributable to the more complicated algorithms required. The pseudo-random key encryption required 20.05 micro-seconds per word, which can be accounted for by the need to generate a 60-bit random number for each word of data. The pseudo-random number generation algorithm was based on the FORTRAN function RANF. Since the 6400 lacks direct 60-bit multiplication, two separate multiply operations were required per data word. Then by means of masking the shifting, a 60-bit random number was constructed. This additional processing certainly required more time

than would be necessary if 60-bit pseudo-random numbers could have been generated directly.

Constant Errors

To interpret the results, the contributions of extraneous factors must be distinguished from the time required by the encryption processes themselves. Among these factors are delays introduced by execution of the clock readout program, and by initialization and termination processes of iteration loops. In order to isolate the effect of these influences, runs were made using blocks of data of varying sizes. Five runs were made in which the length of the message block was varied from 4000 to 20,000 words. In all cases, the processing time per word showed non-systematic variation of less than 0.3μs as the number of words increased, indicating that the constant contribution from clock readout delay and loop initialization and termination control was not significant.

Intermittent Errors

In addition to constant factors distorting the time measurements, variable delays may also occur. In particular, when a clock readout is requested by an active job on the 6400 CPU, a flag is set to notify the operating system of the request. But the operating system, which runs in a separate peripheral processor, may not respond immediately, and indeed the delay period may reach a millisecond before the operating system reports the current clock reading to the requesting program.

Furthermore, jobs on the CPU are swapped at least three times per second, so that after about 0.3 seconds of execution, a job is removed from the CPU and all parameters are recorded. In recording the clock

parameter, some error is likely, which in turn would further distort measurement of the process being timed.

To determine the range of these variable errors, 100 separate tests were made encrypting the same 20,000 word text. The mean totals and standard deviations of the 100 runs were calculated, as well as the mean averages for encrypting a single word. If we assume that the variable errors followed a normal distribution, we can determine the confidence interval for the time measurements using the t-distribution. The probability is .95 that the true mean time falls within the range $M \pm 1.98\ \sigma$, where M is the sample mean and σ is the sample standard deviation. In the case of the null encryption, $M = 0.09526$ and $\sigma = .0009962$ seconds. Thus, there is a probability of .95 that the actual time required by the null encryption falls with the range $0.09526 \pm (1.98 \times .0009962 = .0020)$ seconds. Hence the time for processing a single word by the null encryptinon can be determined to be 1/20,000th of this figure, or $4.76 \pm .10\mu s$. Similarly, the time to encrypt a word of data using the one-word key is $4.78 \pm .11\mu s$, using the long key is $8.25 \pm .11\mu s$, using the double key is $12.56 \pm .14\mu s$, and using the pseudo-random key is $20.05 \pm .15\mu s$. All these figures have a confidence level of 95%.

Comparison with FORTRAN Results

The encryption routines were also programmed in FORTRAN for purposes of comparison. 6400 FORTRAN, however, lacks the exclusive-or function, which is the heart, so to speak, of additive encryption. The function might be realized by the Boolean equivalence,

$$x \oplus y \equiv (x \wedge y) \vee (\bar{x} \wedge \bar{y}).$$

But the corresponding FORTRAN expression,

(X .AND. Y) .OR. (.NOT. X .AND. .NOT. Y),

is particularly inefficient, requiring five in-line function invocations. Therefore it was judged not to be suitable for a test of processing time requirements, and instead an assembly language enclusive-or subroutine called LG01XR was used.

The same series of tests were conducted using the FORTRAN routines as had been run in assembly language. The times required by the FORTRAN processes were about two to four times greater than required by corresponding assembly language processes. The FORTRAN results are summarized in Table II, and the relationships of the time requirements for FORTRAN and assembly code are indicated in Fig. 1.

Encryption Time Coefficient

In order to provide a more or less machine-independent measure of the time penalty imposed by encryption methods, an encryption time coefficient is introduced, defined to be the ratio of the time required to encrypt data versus the time required simply to fetch and store those data without modification (the null transformation). One word key encryption in assembly language has an encryption time coefficient of 1.00, indicating that there is practically no time penalty imposed by this method. The worst assembly language case, pseudo-random key encryption, has a coefficient of 4.21, indicating that it is about four times slower than simply fetching and storing. The coefficients are significantly higher for the FORTRAN routines, ranging from 2.68 for one-word key encryption to 9.96 for pseudo-random key encryption. FORTRAN encryption is thus not only slower in absolute terms than assembly language encryption, but

the ratio of encryption to simply moving data in FORTRAN is proportionally greater than in assembly language.

Conclusion

The time taken to encrypt 20,000 words of data by four additive methods on a Control Data 6400 computer was measured using assembly language and FORTRAN routines. Standard deviations were computed for 100 runs by each method, and from this the range of error at the 95% confidence level was calculated. The time required for selecting and fetching the cleartext data and for storing the cryptotext was separately measured (the "null encryption"), and was then compared with the times required by each encryption method. The following time requirements were determined for the assembly language routines: one-word key encryption required 4.78 $\pm$ .11μs per word or 2,092,000 characters/second, long key encryption required 8.25 $\pm$ .11μs per word or 1,212,100 characters/second, double key encryption required 12.56 $\pm$ .14μs per word or 796,200 characters/second, pseudo-random key encryption required 20.05 $\pm$ .15μs per word or 498,800 characters/second. The same processes required about two to four times more time when programmed in FORTRAN.

Acknowledgement

We are indebted to Professor L. A. Jaeckel of the Statistics Department, University of California, Berkeley, for his helpful suggestions regarding statistical methodology.

REFERENCES

1. Peterson, H. E., and Turn, R., "System Implications of Information Privacy," Proc. AFIPS 1967 SJCC, vol. 30, AFIPS Press, Montvale, N. J., pp. 291-300.
2. Skatrud, R. C., "The Application of Cryptographic Techniques to Data Processing," Proc. AFIPS 1969 FJCC, vol. 35, AFIPS Press, Montvale, N. J., pp. 111-117.
3. Baran, Paul, "On Distributed Communications: IX Security, Secrecy and Tamper-Free Considerations," Doc. RM-3765-PR, RAND Corp., Santa Monica, Ca., 1964.
4. Turn, R., and Shapiro, N. Z., "Privacy and Security in Databank Systems: Measures of Effectiness, Costs, and Protecter-Intruder Interactions," Proc. AFIPS 1972 FJCC, vol. 41, pp. 435-444.
5. Carroll, J. M., and McLelland, P. M., "Fast 'Infinite-key' Privacy Transformations For Resource-Sharing Systems," Proc. AFIPS 1970 FJCC, vol. 37, pp. 223-230.
6. Hoffman, Lance J., "The Formulary Model For Access Control," Proc. AFIPS 1971 FJCC, vol. 39, AFIPS Press, pp. 587-601.
7. Hoffman, Lance J. (ed.), Security and Privacy in Computer Systems, Wiley, 1973 (contains references 1-6).
8. Van Tassel, D., Computer Security Management, Prentice-Hall, Englewood Cliffs, N. J., 1972.
9. Turn, R., "Privacy Transformations for Databank Systems," Proc. 1973 National Computer Conference.
10. Feistel, Horst, "Cryptography and Computer Privacy," Scientific American, vol. 228, no. 5, May, 1973.

11. Golomb, S., Shift Register Sequences, Modern Algebra Series, Holden-Day, 1967.

12. Feistel, H., Notz, W. A., Smith, J. L., "Cryptographic Techniques For Machine to Machine Data Communications," IBM Corp., Yorktown Heights, N. Y., RC 3663, December, 1971.

13. Tuckerman, Bryant, "A Study of the Vigenere-Vernam Single and Multiple Loop Enciphering Systems, "IBM Corp., Yorktown Heights, N. Y., RC 2879, May, 1970.

14. Graff, C., "Costs of Privacy Transformation Using the Formulary Method For Access Control," Research Project For MS Program, Dept. of Electrical Engineering and Computer Sciences, University of California, Berkeley, 1972.

15. Conway, R., Maxwell, W., and Morgan, H., "Selective Security Capabilities in ASAP--A File Management System," Proc. AFIPS, 1972 SJCC, vol. 41, pp. 1181-1185.

16. Garrison, W. A., and Ramamoorthy, C. V., "Privacy and Security in Data Banks," Technical Memo. No. 24, Electronics Research Center, University of Texas, Austin, Texas, 1970.

TABLE I (Assembly Language Routines)

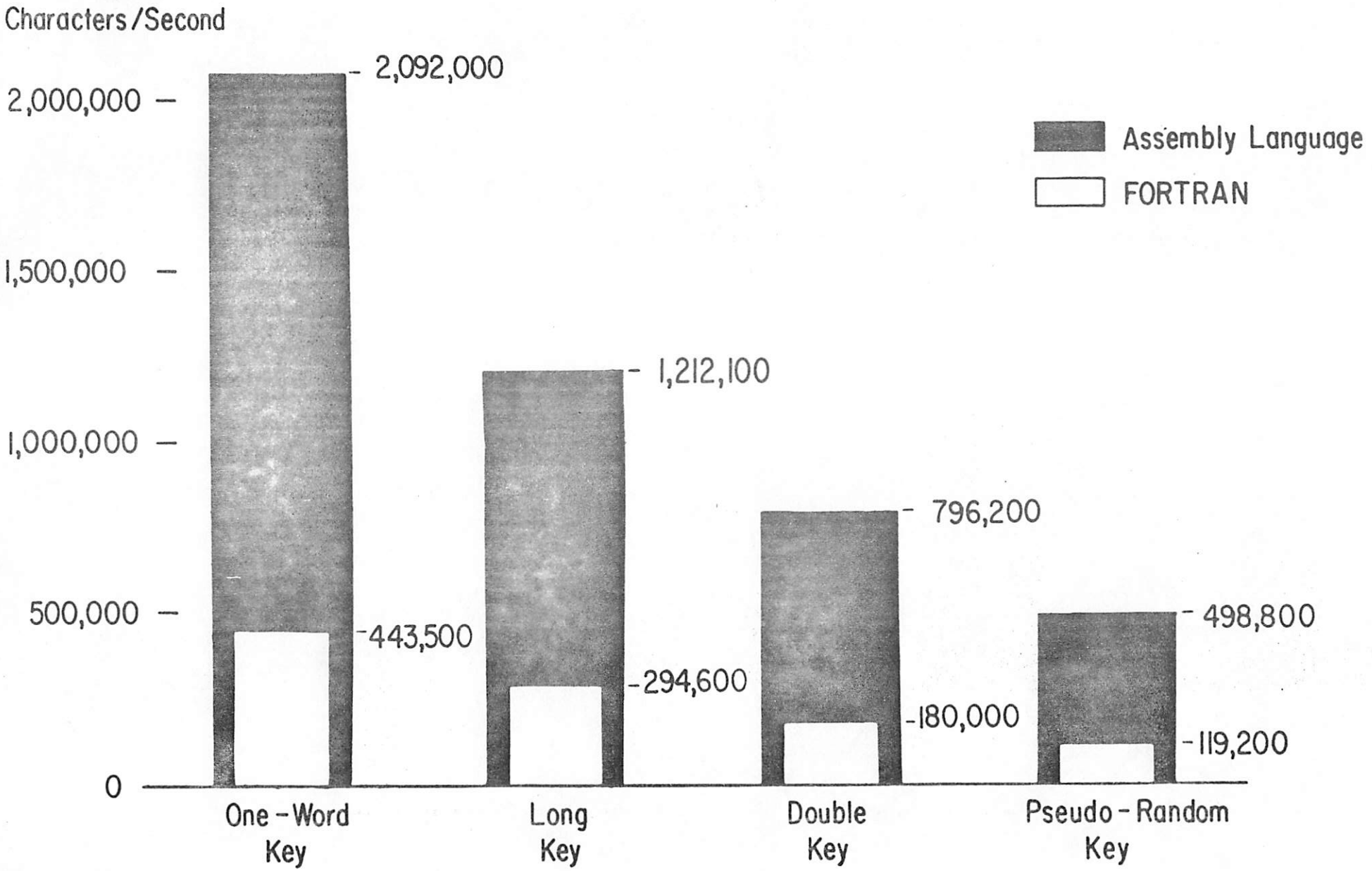
	Time per word (95% confidence level)	Approximate Data Rate (Based on 10 characters per word)	Encryption Time Coefficient (Ratio of encryption time to null tranformation time)
Null Tranformation	4.76 ± .10μs	2,100,800 chars/sec	1.00
One Word Key	4.78 ± .11	2,092,000	1.00
Long Key	8.25 ± .11	1,212,100	1.73
Double Key	12.56 ± .14	796,200	2.64
Pseudo-Random Key	20.05 ± .15	498,800	4.21

CPU time required by CDC 6400 to encrypt data using assembly language routines

TABLE II (FORTRAN Routines)

	Time per word	Approximate Data Rate	Encryption Time Coefficient
Null Transformation	8.42 ± .07μs	1,187,600 chars/sec	1.00
One Word Key	22.55 ± .10	443,500	2.68
Long Key	33.95 ± .12	294,600	4.03
Double Key	55.55 ± .12	180,000	6.60
Pseudo-Random Key	83.86 ± .14	119,200	9.96

CPU time required by CDC 6400 to encrypt data using FORTRAN routines



CDC 6400 ENCRYPTION SPEEDS