SPECIAL FEATURE

Exhaustive Cryptanalysis of the NBS Data Encryption Standard

Whitfield Diffie
and
Martin E. Hellman
Stanford University

Introduction

For centuries, cryptography has been a valuable asset of the military and diplomatic communities. Indeed, it is so valuable that its practice has usually been shrouded in secrecy and mystery.

The growing commercial need for cryptographic systems, however, requires an expansion of public knowledge in this area. In response to this need, IBM instituted a research program in the late 1960's to develop high-grade cryptosystems for use in its product line. It successfully introduced a cash dispensing terminal based on this research, and applications to terminals, tape and disk drives, etc. appear imminent. Because these applications would benefit from a well-defined standard or set of standards, the National Bureau of Standards (NBS) has adopted a data encryption standard[1,2] designed by IBM. To help preserve competition, IBM will not receive royalties on most devices which comply with the standard.[1]

Unfortunately, the proposed standard is too weak for some applications and should be modified. We have attempted to make NBS aware of this problem,[3,4] but our efforts thus far have been apparently unsuccessful. NBS has raised a number of objections to the technical validity of our position. This paper is intended to carefully set down our reasoning so that the technical community can form its own opinion.

The following section provides the basic argument concerning the standard's inadequate level of security. It shows that, using the simplest of cryptanalytic attacks, a $20 million machine can be built to break the proposed standard in about 12 hours of computation time. The equivalent cost per solution is only $5000 (obtained by depreciating the machine over five years). Thus, the proposed standard's level of security against this attack is high today—but not excellent, since major intelligence agencies possess the financial resources and the interest to build such a machine.

More seriously, in about 10 years time, the rapidly decreasing cost of computation will bring the machine's cost down to the $200,000 range, and the cost per solution down to the $50 range. The standard will then be almost totally insecure.

For this reason the standard may have to be replaced in as few as five years—just as it is coming into widespread use. The cost, inconvenience, and loss of compatibility associated with this planned obsolescence can be avoided by making minor modifications to the standard which incur essentially no additional cost.

While NBS disputes our claims, it has indicated that changing technology will probably cause the standard to be revised in five or 10 years.[5] Equipment which makes use of the standard should be designed to minimize the cost of substituting a new standard.

There is also the danger that encrypted messages which are too costly to break now will be stored and broken in a few years at a much lower cost. While most confidential data have short privacy time constants, today's medical records, income tax returns, census data, etc. should still be private 10 years from now.

The basic argument

In this section we develop the basic argument concerning the standard's level of security. The following two sections are primarily a justification for the assumptions of this basic argument.

The proposed standard transforms a block of 64 plaintext (unenciphered) bits, denoted P, into a block of 64 ciphertext bits C. This transformation is governed by a 56-bit key K, and is invertible so that

$$C = S_K(P)$$
$$P = S_K^{-1}(C)$$

where S_K is the enciphering transformation when key K is used. There are 2^{56} keys.

We will consider a known-plaintext[6] cryptanalytic attack in which the cryptanalyst has several corresponding plaintext-ciphertext blocks, all encrypted in the same key. On the basis of these he tries to determine the key for use in reading later cryptograms for which he does not

know the plaintext. Or, he may use the key to inject properly enciphered messages of his choosing into the system to foil its authentication aspects. The known-plaintext attack is more formidable than the ciphertext-only[6] attack familiar to enthusiasts of puzzle ciphers, in which the cryptanalyst has intercepted a quantity of ciphertext and has only partial knowledge of the structure of the plaintext (e.g., it is in English so "E" occurs 13% of the time, etc.).

We consider the known-plaintext attack for several reasons:

1. Most successful, professional cryptanalysis is based on variations of the known-plaintext attack. Especially in a commercial system, it would be impractical to require that old plaintext be kept secret or paraphrased if declassified (e.g., timed press releases).
2. A later section indicates how the attack can often be successfully modified to a ciphertext-only attack.
3. NBS has agreed that the system should be secure against a known-plaintext attack.[7]

The attack described in this paper is based on brute force, but it would be successful. Let P and C denote a known plaintext-ciphertext pair related by the unknown key K. Decipher C under each of the 2^{56} keys until one is found which yields the known plaintext P. With minor exceptions (discussed later), this key equals K.

Such a search of the key space might seem infeasible because there are $2^{56} \doteq 10^{17}$ keys, so even if one key could be tried each microsecond, it would take 10^{11} seconds, or about 10^6 days, to do an exhaustive search. A million devices searching in parallel, however, would take only one day for an exhaustive search and one-half day for the average search, since the solution is found after trying one-half the keys on the average. Because the standard was chosen so that it could be implemented on a single LSI chip, a million-fold parallel processor with 10^6 IC's is conceivable, although it certainly requires further justification on technical and economic grounds.

Continuing to use order of magnitude estimates, in million quantities, the chips could be bought for about $10 each. Allowing a factor of two for design, control hardware, power supplies, PC boards, racks, etc. results in a $20 million machine. Depreciating this cost over five years yields a daily operating cost of $10,000, which translates into an average cost of $5000 per solution. Since the cryptanalysis yields the key in use, all material enciphered in that key is compromised. If, for example, an organization's personnel records are all enciphered in the same key, the cost per record is much smaller than $5000.

It should be reemphasized that the above estimate was only accurate to order of magnitude, and we would not be surprised if today it would cost $5 million to $50 million to build the machine. However, as discussed later, this machine will benefit fully from the decreasing cost of hardware and computation which has fallen about an order of magnitude every five years since the 1940's. Thus even a factor-of-10 error would be erased in five years. The remaining sections indicate, though, that even a more accurate estimate is still in the $20 million range. The probable error in this estimate is about a factor of two.

The decreasing cost of computation has an even more serious effect. In 10 years this $20 million machine should be a $200,000 machine and a half day's time will cost only $50, although design costs may double these figures. Both the initial investment and the cost per solution will then be much too small to offer an adequate level of security. NBS has acknowledged that changes in the standard are planned to meet advances in technology.

Since the standard will be included in terminals, disk drives, tape units, etc. in hardware form, modifying the equipment to accept a new standard will be relatively expensive. If design constraints permit and physical security is adequate, using a plug-in module would minimize the changeover cost. The revised standard will probably have a larger key, and buffers should be designed accordingly.

Changing the standard will diminish its usefulness in fostering compatibility and interconnection. While the new standard will probably be upward compatible, there is the problem of reading new data with an old terminal and of distinguishing old from new data.

This planned obsolescence is unwarranted, and it is easily remedied by increasing the key length from 56 bits to 128 or 256 bits. Use of a 128-bit key would increase the estimated cost for a brute force search from $5000 to $\$2 \times 10^{25}$, and no foreseeable technological advances would allow this to be brought into a reasonable range. Indeed, quantum mechanical and thermodynamic[8,9] considerations rule out exhaustive searches on keys of several hundred bits.

It is important to note that while too small a key guarantees insecurity, a large key does not necessarily guarantee a high level of security. There may be shortcuts which allow successful cryptanalysis in much less time than is required by exhaustive search. For example, a monoalphabetic substitution cipher has $26! \doteq 4 \times 10^{26}$ keys, yet it is quickly solved by hand from frequency counts on letters, pairs of letters, etc. Similarly even a 128-bit version of the standard may succumb.

Several other studies of the standard have been performed. NBS held a workshop[10] in August 1976, which addressed the possibility of exhaustive search. Its conclusion, in complete disagreement with this paper, was that current technology would not support a machine of the type proposed herein. Rather, at the earliest, it would be 1990 before such a machine could be built.

IBM performed a separate study which concluded that it could deliver such a machine by 1981 at a price of $200 million. This is within an order of magnitude of our $20 million cost estimate, and is even closer when the difference between manufacturing cost and price is considered. (IBM has since disavowed the conclusions of this study and has taken the position that the conclusions of the NBS workshop should be accepted.) We hope that this paper will clafify our reasons for adhering to our position.

The authors, with several others, conducted a short (one month) study[11] which looked for structure and potential weaknesses in the standard. A symmetry under complementation was found which allows a 50% savings in search effort under a chosen plaintext attack. Potential weaknesses were also found and methods for removing them were suggested.

This study suffered because the government has asked IBM to keep secret the structures designed into the memory tables used by the Data Encryption Standard (DES). We think it is unreasonable for a public security standard to have secret structures because, if someone involved in the design of the standard were to turn against it, he would be in a much better position to break it. For this reason, it is a well-established principle that the security of the cryptosystem should not depend on secret design principles. We encourage NBS to heed this principle and to press for the public disclosure of all structures and design principles used in the standard.

Both IBM and NSA have performed evaluations of the security level offered by the standard. We also encourage NBS to seek the public release of these studies to help

remove doubts that significant savings over an exhaustive search are possible.

Morris, Sloane, and Wyner of Bell Laboratories have also evaluated the standard.[12] They find it wanting and make a number of suggestions for improving it, including enlarging the key size.

Objections to the basic argument

When we presented NBS with the basic argument of the preceding section, it responded[13] with a number of objections which culminate in the conclusion that it would take 91 years, not one day, to do an exhaustive search. (More recently, Dr. Ruth Davis of NBS used a 2000-year estimate.[14]) NBS bases this estimate on the assumption that, at most, 1000 parallel search devices can be used and that 40 μsec is the fastest that a key can be tried. These assumptions increase the search time by factors of 1000 and 40, respectively, for an overall increase to 40,000 days, which is approximately 91 years.

This section discusses eight objections which are claimed by NBS to invalidate our argument.

Design and control costs

Objection: The design and control costs overshadow the CPU hardware costs in a parallel processor. These costs grow much faster than linearly in the degree of parallelism.

This is true for a parallel processor such as the Illiac IV where the processors must interact. However, the architecture we envision, discussed in the next section, is more closely related to a large semiconductor memory than to a parallel processor. The repetitive structure, low I/O volume, and lack of component interaction greatly simplify the design, including automatic fault diagnosis. Our conclusion is that, today, design and control costs will not greatly add to the total cost.

MTBF

Objection: The mean time between failures in a million-component system would be much less than one day. The machine would hardly ever complete an error-free search.

LSI IC's are typically specified to have a failure rate of 0.05% per 1000 hours, but frequently have actual failure rates of 1% per 1000 hours. Using the conservative 1% per 1000 hours figure, a machine with 1,000,000 devices has an MTBF of 0.1 hour or 6 minutes. By cooling the machine room it should be possible to obtain the 0.05%-per-1000-hour failure rate which would correspond to a two-hour MTBF.

These MTBF's do not present a problem because, as detailed in the following section, the machine would be built in 64 racks, each with about 16,000 components. Each rack would be capable of detecting its own chip failures and of signaling a minicomputer controller to switch over to a spare rack while the fault was being repaired. Repairing a rack would consist of replacing the card with the failed chip and could be accomplished in less than 10 minutes. Therefore, only three or four spare racks are needed to ensure essentially continual, error-free operation.

Speed and cost

Objection: It is not possible to build an LSI chip to test a key in 1 μsec for $10. Rather, 40 μsec and $100 are needed.

Another section discusses the chip design and speed and shows that 1 μsec is a reasonable estimate of the time required per key with 1977 technology.

These high speeds are possible because of the very small amount of I/O required. NBS is apparently using speed figures for an MSI TTL implementation which they used as a test bed. This device takes 20 μsec to load a 64-bit message and 20 μsec to unload the resultant 64-bit cryptogram so that I/O alone takes 40 μsec. In addition, the device takes 13 μsec for computation. The dominance of I/O time makes lowering this computation time of little value.

The cryptanalytic search machine will not use the standard data encryption chip, but rather will utilize a special search chip which is optimized for computation speed at the expense of I/O speed. This is economical because of the extremely low I/O volume. Neglecting diagnostic testing, each chip need only be loaded at the beginning of the day with 184 bits (64 bits of plaintext, 64 bits of corresponding ciphertext, and 56 bits to specify the starting point in the key space for the chip's search). The output is even more limited. Most chips will have no output unless they fail and output an improper solution. (These failures are easily detected by checking each supposed solution to see if it really does decipher the known ciphertext into the known plaintext.) And the one chip which is searching the portion of the key-space containing the right key will have one 56-bit output. All other operations, including incrementing the key, are done on the chip.

The low volume of I/O allows serial input to be used, reducing the pin count requirement to only seven active pins, although a 16-pin package may be required because of the large die size. Pin count is a factor in determining an IC's cost, and the cost of such a chip in million quantities would be approximately $10.

NBS's $100 per chip figure is reasonable for small quantity purchases with several markups.

Physical size

Objection: a million-chip machine would require 6000 six-foot-high racks.

NBS bases this estimate on the assumption that 20 in² (129 cm²) of board area are needed per search chip. They also assume each search chip needs one Intel 8080 or similar microprocessor for interfacing and control.[13]

The low pin count allows a density of at least one search chip per in² (6.5 cm²); 128 search chips can thus be mounted on an 11″ × 13″ (28 cm × 33 cm) printed circuit board with room left for up to 15 packages of control logic. The low I/O volume allows use of only one interface-control unit per board, timeshared among the 128 search chips on a polling basis.

A million-chip machine thus requires only 64 racks, each with 128 such cards.

Power requirements

Objection: A million-chip device operating at high speed would consume too much power.

Unless special precautions are taken, LSI IC's are limited to about 1 watt of power dissipation. Because there is a linear tradeoff in speed and power, the question becomes whether 1 watt is sufficient power to allow a 1 μsec search time per key.

As will be shown in a later section, the high-speed portion of the chip includes about 3000 gates. To obtain a 1-μsec search time, these must operate with approximately a 4-nsec gate delay. The equivalent number of

gate operations per second is $3000/(4 \times 10^{-9}) = 8 \times 10^{11}$, which requires a speed-power product of at most 1.2 picojoule-per-gate operation for the power per chip to be less than 1 watt.

Speed-power products of 0.2 pJ are obtainable with CMOS/SOS (complimentary metal oxide semiconductor/silicon on sapphire) technology.[15,16] The less expensive N-channel Si-gate depletion load technology is also a contender. In 1974 it had a speed-power product of 38 pJ,[15] while the Intel 2125/15 memory introduced in 1976 had an 8 pJ speed-power product, and memories introduced in 1977 are expected to reach 2 pJ. Further, these are static power figures. The search chip's dynamic speed-power product would be lower. Both CMOS/SOS and N-channel IC's meet the other requirements of high speed and density.

Using a 1-watt/chip figure and allowing a factor of two for power supply losses, cooling, control logic, etc., we find that 2 megawatts of AC power are required by the machine. At $0.03 per kW-hr this costs $1500 per day, which is low compared to the $10,000-per-day amortization cost. Use of a low-power technology (CMOS/SOS) would reduce these figures by a factor of five.

Cost of larger key

Objection: Increasing the key size from 56 to 128 or 256 bits would greatly increase the cost of the device.

We will see in a later section that the encryption chip's area requirements are dominated by elements whose number is independent of the key size. Key-related registers, etc. make up less than 10% of the chip's device count. Going to a 128-bit key would therefore increase chip size by at most 20%. Since packaging would be unchanged, the increase in the chip's cost would be minimal. A 256-bit key may involve a more significant cost at this point in time.

A larger key is also well within the storage limitations of a magnetic card, on which it is envisioned most keys will be stored. If, however, the user must memorize and type in his key, there is a human-factors cost to be considered.

While techniques for remembering long keys can be developed, if a 128- or 256-bit key is adopted in the standard, there will undoubtedly be some applications where shorter keys will suffice and be desired. We therefore suggest that a variable key size be part of the standard.

This is best done by designing the algorithm to use a key as long as the largest key needed, and building in a key expansion algorithm to make shorter keys compatible with the basic algorithm. (The currently proposed standard, in fact, already has a key expansion algorithm because it needs 48 keying bits in each of 16 iterations of a basic enciphering operation. These 16 x 48 = 768 keying bits are obtained by repeating the 56 key bits about 14 times each.) Another section describes two variable-key-size techniques.

In summary, the increased cost of using a larger key is negligible, and changing over to a larger key need not greatly delay introduction of the standard since most of the algorithm is still usable. Only the key expansion operation needs to be modified.

Changing keys

Objection: Cryptanalysis by exhaustive search can be nullified by frequently changing keys.

In the limit, if the key is changed with each block, the system is essentially a one-time pad which is known to be secure against any cryptanalytic attack.[18,19] However, the key distribution problem is horrendous and prevents

Use order form on p. 127.

use of the one-time pad system in all but very special applications, such as the Washington-Moscow hotline.

Rather, NBS must be thinking that if the key is changed every hour, cryptanalysis will buy less information than it would if the key were changed daily. Further, if the information obtained is of short time value (e.g., one hour), a one-day search time would yield only stale data.

But a national standard must be usable in diverse applications, most of which do not allow frequent key changes. Further, one hour or even one minute's information can be worth $5000 in applications such as finance. Even if an exhaustive search takes longer than the time between key changes, partial searching can still be a threat. If keys are changed hourly and learning the key in use is only of value during that time, a machine which takes one day to do an exhaustive search has one chance in 24 of discovering the key during its useful period. On the average, one such key will be found per day. The cost per solution is thus $10,000, only double the usual cost. It is primarily the useful time duration of the solution which differs.

Uniqueness of solution

Objection: There is no guarantee that a single plaintext-ciphertext pair will uniquely determine the key.

This is a difficult objection to treat rigorously since proving the uniqueness of the solution key is probably more difficult than cryptanalysis. Even so, informal reasoning convincingly removes this objection.

First, note that the 64 plaintext-ciphertext bits give us 64 nonlinear equations in 56 unknowns (the key bits). The form of the equations (i.e., the design of the algorithm) would have to be carefully chosen if there are to be multiple solutions. For example, if the equations were linear with randomly chosen coefficients, then the probability of there being multiple solutions would be less than one in 100. (This is derived by noting[20] that if k binary n-vectors span an m-dimensional space and a k + 1st vector is chosen uniformly at random, then the probability that it is independent of the first k vectors is $(2^n - 2^m) / 2^n$ for $m < n$.) If there were multiple solutions, there would probably be only two.

Further, even if there were as many as 1,000,000 solutions, use of additional plaintext-ciphertext equivalents would allow rapid determination of the one correct solution. And even with 1,000,000 solutions, the amount of I/O is minimal since each chip would find one solution per day on the average.

Our conclusion is that none of these objections are really valid, and that the 56-bit key is too small for a national standard cryptosystem. The next two sections provide justification for a number of the statements made in this section.

System architecture

To avoid MTBF problems and to minimize design and control costs, the machine has 64 search racks, operating almost autonomously, all under the control of a minicomputer. There are three or four spare racks to provide backup for failed units which are being repaired. Each rack contains 128 printed circuit cards with 128 search chips on each. The total number of active search chips is thus 64 x 128 x 128 = 2^{20} = 1,048, 576 or slightly more than the 1,000,000 used for the basic argument. Each rack has one or two additional cards for interfacing and controlling the 128 search cards. Each search card has several IC packages for interfacing and controlling its 128 search chips.

Each of the three levels of control uses a common I/O bus for all units under its command, and has an X-Y type polling structure. For example, the 128 search chips on a card are arranged in a rectangular array of 8 x 16 chips. Each chip has a pair of pins for sensing when it is being addressed, and the controller addresses the $(i, j)^{th}$ chip by driving the ith of 8 horizontal wires and the jth of 16 vertical wires.

To initialize the search, the minicomputer addresses each of the 64 racks in turn and transmits a reset command followed by 184 bits (64 bits of known plaintext, 64 bits of corresponding ciphertext, and 56 bits to specify the starting point for the rack's search of the key space). Once the control unit within the rack has this information, it addresses each of its 128 search boards in turn. Each board's control unit is reset and given the same 184 bits as received by the rack's control unit except the starting key is incremented by 2^{43} for each new board since each board searches (1/64) (1/128) = 2^{-13} of the key space which is 2^{43} keys. The board's control unit then addresses each of its 128 search chips in a similar manner except the starting key is incremented by 2^{36} for each new chip. This technique eliminates what could be a horrendous fan-out problem.

Once in the search mode, polling is done in a similar manner to avoid fan-in problems. Each board's control unit polls its 128 search chips on a regular basis. Any solutions within a board are relayed to the rack controller when it polls that board. And finally, any solutions within a rack are relayed to the minicomputer when it polls the rack.

Automatic fault diagnosis is built in by having the board controller check any solutions offered it. This is done by having an extra search chip on each board. The known ciphertext and the supposed solution key are entered, and the computed plaintext is compared with the known plaintext. Any disagreement indicates a chip failure and causes the replacement of the entire rack by a spare while the defective board is replaced. For reasons to be developed shortly, the replacement rack can take up the defective rack's search almost at the point where the failure occurred. Similar checks on solutions are built in at the rack-controller and minicomputer level to guard against failures in the lower-level controllers.

These checks will automatically diagnose and correct a fault which causes a false alarm, but there is a need to detect and diagnose a fault which would cause a valid solution to be lost. This is accomplished by having each rack go into a check mode every 30 minutes. In this mode, each chip within the rack is given the same plaintext-ciphertext pair and the same starting point for its search. This data is not related to the actual search but, rather, is chosen so that the solution will be found after trying about 10 keys. Each chip should produce the same solution when polled. By doing this about 10 times with different data, essentially every gate in each chip will be exercised and tested. Using a 200-kbps I/O bus, it will take the rack about 0.5 msec per chip per test, 8 seconds per test (there are 16,000 search chips in a rack), and 1.5 minutes for 10 tests. Because this is done every 30 minutes, only a 5% speed penalty is incurred. At even a 1%-per-1000-hours chip failure rate, the probability of a rack experiencing a failed chip in a 30-minute period is only 0.08.

Before entering the check mode, the rack notes the number of keys searched by its slowest chip, and if no failures are found, takes up the search at that depth when the check is completed. The rack is able to keep track of this slowest chip because of the polling method employed. When a chip is polled by its board control

Table 1. Characteristics of major microprocessor technologies.

TECHNOLOGY	SPEED POWER PRODUCT (pJ)	GATE PROPAGATION DELAY (NSEC)	GATE DENSITY GATES/MM²	DATE OF INITIAL PRODUCTION
P-CHANNEL METAL GATE	450	80	50	1966
P-CHANNEL SI-GATE	145	30	90	1969
SCHOTTKY BIPOLAR	60	6	25	1969
N-CHANNEL SI-GATE (HIGH VOLTAGE)	45	15	95	1972
N-CHANNEL SI-GATE DEPLETION LOAD	38, 8, 2	12, 4, 2	110, 150, 170	1974, '76, '77
SI-GATE CMOS	0.5	10	45	1973
I^2L	1	50	40	1975
CMOS/SOS	0.2	3	100	1977

unit, it outputs one bit to indicate if it has found a solution, followed by the last key tried. If the first bit indicates that a solution has been found, the solution is the last key tried. Because it is the low-order 36 bits which are searched by a chip, the board controller compares the low-order 36 bits of the last key tried with the minimum value found thus far in its current polling of the 128 search chips. At the end of the polling cycle, the minimum value found is relayed to the rack which keeps track of the minimum value among all its boards. This value is used for reinitializing the search after a check mode test or after a failure is detected. Since the only differences in chip search rate are due to a chip finding a solution, little is lost by using this "conservative" measure of search depth.

Each rack and board has three modes of operation—reset, search, and check—while each chip has only two modes—reset and search. The chip search mode is used by the board in its check mode. Keeping the lowest control level simple makes economic sense because there are over a million chips.

It is now seen that only seven active pins are needed: ground, power (+5v), I/O, control (reset vs. search mode), X and Y sensing (for polling), and a clock pin. The board layout is simplified because all of these pins, except the X and Y sensing pins, are connected to common busses on each board, and even the X and Y sensing pins have a simple, repetitive connection schedule.

This description of the system architecture should also make it clear that the high degree of parallelism and the large number of components present little trouble. Indeed, the 64 racks could operate totally independently with manual setup and readout.

Chip design

The power, gate density, and speed requirements point to CMOS/SOS or N-channel depletion load as the technology to be used, although I^2L (integrated injection logic) cannot be ruled out at this point.

The need for low speed-power products (approximately 1 pJ per gate operation) has already been established above. A high-density technology is also needed since the chip has about 6400 devices, comparable to Zilog's Z-80 microprocessor. A density of at least 100 gates/mm² is needed. Also, a gate delay of about 4 nsec is needed to achieve a search time of 1 sec per key.

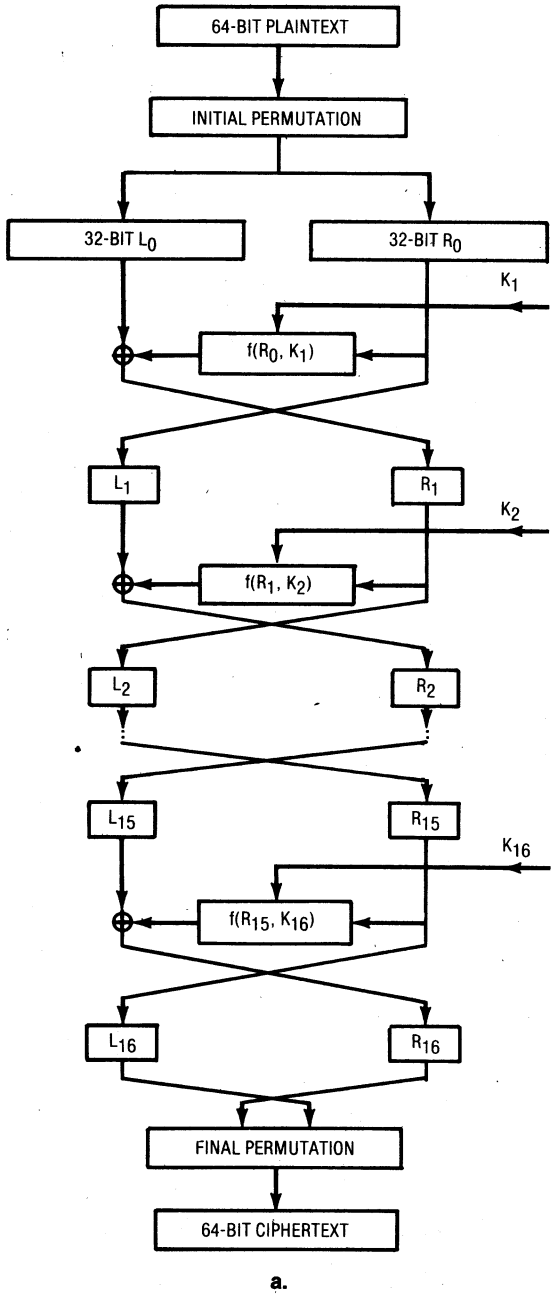
Table 1, taken primarily from Faggin[15] shows CMOS/SOS to be the most attractive technology for all of these requirements, but indicates that the initial production date is uncertain. In private conversations with several IC experts, we were told that CMOS/SOS is available today for the type of application envisioned in this paper. In these conversations, CMOS/SOS speed-power products of 0.1 to 1 pJ and gate delays of 1 to 5 nsec were quoted, depending on the specifics of the chip's circuitry. Cost estimates of $5 to $20 per chip in quantities of 1,000,000 were also quoted. All of these figures are well within the estimates used in or implied by the basic argument.

These conversations also showed that I^2L had improved considerably since the writing of Faggin's article. Gate delays of 4 nsec and densities of 100 gates/mm² have been achieved in the laboratory and should soon be available in production units. I^2L, therefore, is also a contender.

Two entries in Table 1 were added on the advice of one of the reviewers. He noted that the Intel 2125/15 RAM, introduced in 1976, showed substantial improvement over earlier N-channel depletion load devices in all three characteristics (8 pJ speed power product, 4 nsec gate delay, and 150 gates/mm²), and that further improvements (to 2 pJ, 2 nsec, and 170 gates/mm²) are expected in products to be introduced in 1977. A 2 pJ speed-power product would cause the search speed to decrease to one key every 1.6 μsec and entail a 60% increase in our estimates (to $32 million for the machine and $8000 per

solution). Aside from being well within the order of magnitude estimate that is needed, the speed-power product for a static device is higher than for devices which are constantly processing. The exact savings depends on particulars of the chip layout, but savings of as much as 90% (to 0.2 pJ) might be possible. The low cost and large production experience of N-channel make it an attractive choice.

To understand the density and speed requirements, one must explore the NBS encryption algorithm in greater detail. Figure 1a is a flow chart representation of the algorithm in the enciphering mode. (The deciphering mode is of the same structure). The 64-bit plaintext block undergoes an initial permutation of its bits and then goes through 16 iterations of a key dependent transformation to produce a 64-bit "preoutput" block. This preoutput block undergoes a final permutation to produce a 64-bit ciphertext block.



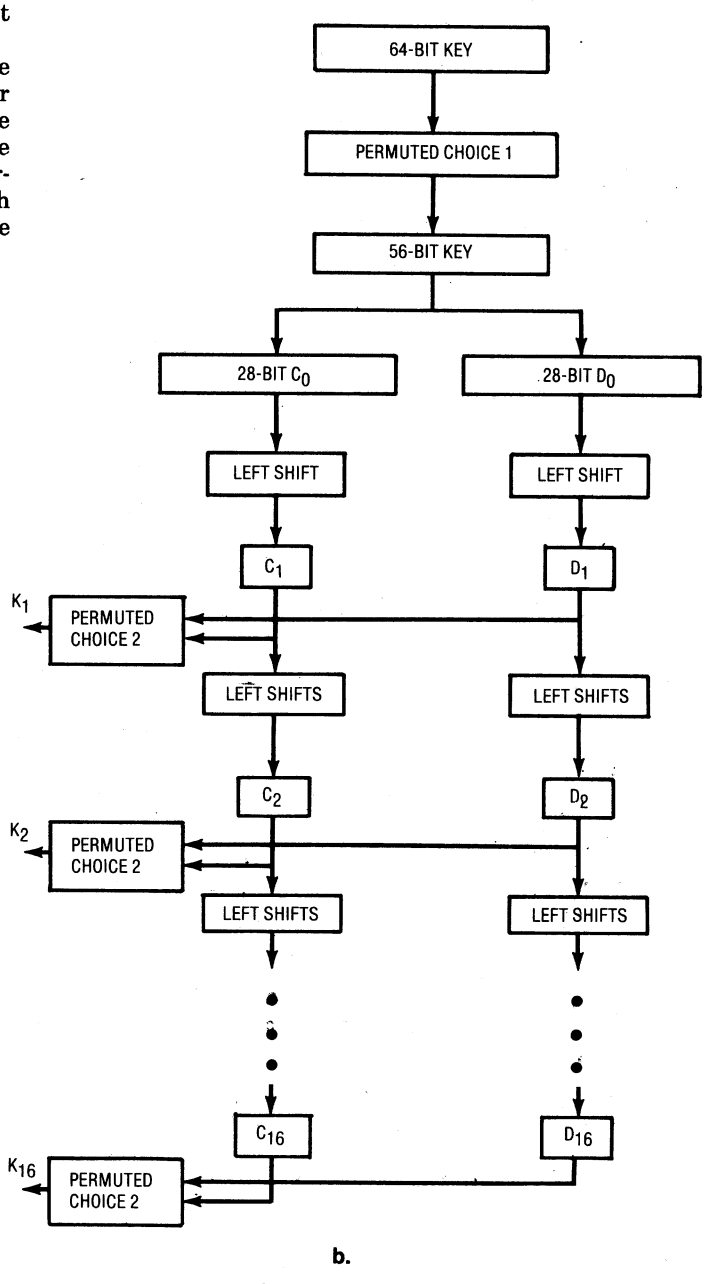


Figure 1. Flowcharts for (a) enciphering and (b) key scheduling algorithms.

The initial and final permutations should be eliminated from the standard since they possess no cryptographic value and take about 20% of the enciphering time in a software implementation. However, for now, we must include them.

The basic loop which is iterated 16 times is of the form

$$L_i = R_{i-1}$$
$$R_i = L_{i-1} \oplus f(R_{i-1}, K_i)$$

$i = 1,2, \ldots, 16$ where K_i is a set of 48 key bits selected from the key according to the key scheduling algorithm; L_i and R_i denote the left and right halves of the 64-bit block after the ith iteration; and $\oplus$ denotes the exclusive or operation. The key scheduling algorithm is represented by the flow chart of Figure 1b and the function of $f(R, K)$ is represented by the flow chart of Figure 2.

With reference to Figure 1b, the 64-bit key first undergoes an initial permutation (permuted choice one) which discards the eight key bits numbered 8, 16, 24, 32, 40, 48, 56, and 64, resulting in a 56-bit key. The eight discarded bits are reserved for parity. While, at first, it seems reasonable to discard the parity bits, closer examination of the key expansion algorithm indicates that there is no cryptographic reason for discarding them. And if all 64 key bits were used in the algorithm, users would have the option of forsaking an internal parity check for a higher level of security. A true 64-bit key would increase the cost of exhaustive search by a factor of 2^8 from $5000 to $1,000,000. Since permuted choice one has negative cryptographic value, we recommend it be eliminated.

Following this initial permutation, the remaining 56 key bits are loaded into two 28-bit shift registers labeled C and D. The contents of these registers at the ith iteration are denoted C_i and D_i. C_i and D_i are obtained from C_{i-1} and D_{i-1} by cyclically shifting the registers either one or two positions to the left. The number of left shifts (one or two) at each iteration is a fixed part of the algorithm which we call the shift schedule. K_i is obtained from C_i and D_i by choosing 48 of the 56 available bits. This choice (permuted choice two) is fixed and the same for all iterations. This completes the description of the key scheduling algorithm.

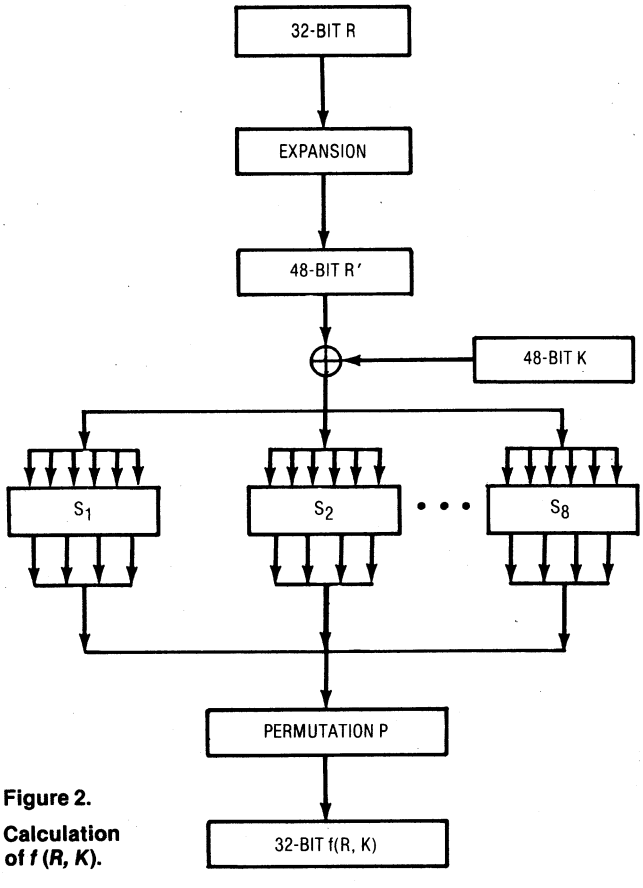


Figure 2.
Calculation of *f* (*R*, *K*).

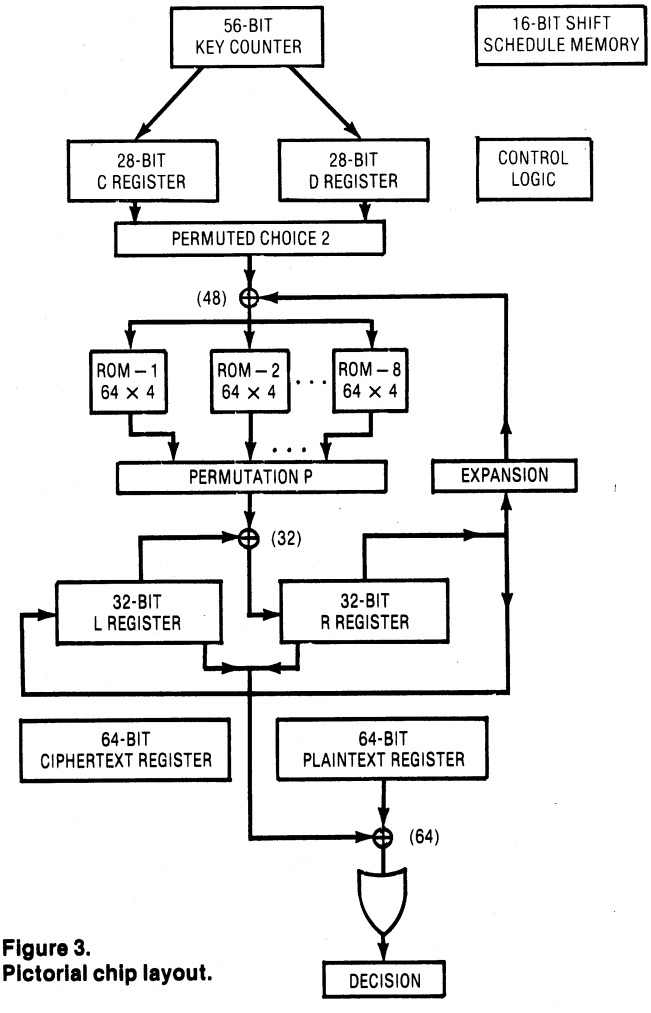


Figure 3.
Pictorial chip layout.

With reference to Figure 2, $f(R, K)$ is computed as follows. First R is expanded into a 48-bit R' by repeating certain bits.
If

$$R = r_1 r_2 r_3 \ldots r_{31} r_{32}$$

then

$$R' = r_{32} r_1 r_2 r_3 r_4 r_5, r_5 r_6 r_7 r_8 r_9 r_{10}, \ldots, r_{28} r_{29} r_{30} r_{31} r_{32} r_1.$$

That is, R is broken into eight 4-bit groups and the last bit of the preceding group and the first bit of the following group are added to each group, resulting in eight 6-bit groups. These 48 bits are XOR'd with the 48 bits of K and successive 6-bit groups are input to eight different ROMs, each of which outputs four bits. After another permutation, P, these 32 bits are taken as $f(R, K)$.

The exact choices for the various transformations (i.e., initial permutation, permuted choices one and two, shift schedule, S_1 through S_8, and the permutation P) are not important to the main purpose of this paper. They are described in the Federal Register[1] and in the document, "FIPS PUB 46, Data Encryption Standard," available from the National Technical Information Service, 5285 Port Royal Road, Springfield, VA 22161, for $3.50.

It is now possible to draw a crude layout of the special-purpose search chip and its timing diagram in the search mode. Figure 3 is a pictorial layout of the chip. The 56-bit key is stored in a register which is also an up counter to facilitate incrementing the key after an unsuccessful test. Permuted choice one can be dispensed with since a solution found by this chip can be transformed into the actual solution by operating with the inverse of permuted choice

one. Similarly, the initial permutation need not be applied to the plaintext and its inverse need not be applied to the ciphertext since the minicomputer controller can account for these two operations by providing properly permuted versions of the known plaintext-ciphertext pair to the search racks.

The 16-bit shift-schedule memory determines whether the C and D registers are to be cycled once or twice, and the control logic includes I/O multiplexers, polling and mode sensors, reset logic, etc.

Permuted choice two is implemented as a set of 48 permuted "wires" as is the permutation P. The expansion operation from R to R' is also of this form except the permutation is particularly simple. The eight S boxes (6-bit to 4-bit mappings) are implemented as eight ROMs with 64- x 4-bit organizations, for a total of 2048 bits of ROM.

The two XOR operations involved in computing $f(R, K)$, and the XOR operation for checking if the computed plaintext equals the known plaintext, are each implemented in parallel with 48, 32, and 64 XOR gates, respectively (these numbers are indicated in parentheses in Figure 3). If any of the outputs of the last 64 XOR gates is one, then the key tried was not correct and the chip tries the next key. If all outputs of these gates are zeros, indicating complete agreement between the computed and known plaintext, the chip goes into an idle mode until polled by the board controller. Since the key counter is incremented as soon as the C and D registers are loaded (see Figure 4), the numerical contents of the key counter will be the solution key plus one.

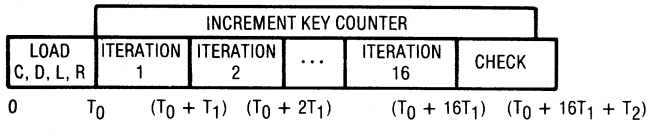


Figure 4. Overall timing diagram.

The timing shown in Figures 4 and 5 is designed to minimize power consumption and compute time per key. For example, instead of incrementing the key counter after checking whether the computed and known plaintexts match, this is done during the 16 iterations and check. Similarly, the C and D shift schedule (SS) registers are shifted during the portion of each iteration when their contents are not needed.

At this point it is possible to do a device count and timing estimate. Table 2 lists estimates of device count for each major "component." It is seen that 6400 devices is a reasonable estimate. This is comparable to the complexity of a Z-80 microprocessor and thus requires a high-density technology.

Turning to the question of speed, let us assume a basic gate delay of 3 nsec, which is typical for CMOS/SOS and which is midway between the 4 nsec of 1976 and the 2 nsec of 1977 N-channel devices. The time for loading the C, D, L, and R registers (T_0 in Figure 4) and for checking if the computed and known paintext match (T_2) is minimal compared to the time ($16T_1$) required for the 16 iterations of the basic loop

$$L_i = R_{i-1}$$

$$R_i = L_{i-1} \oplus f(R_{i-1}, K_i).$$

Figure 5 is used to estimate T_1. An XOR operation involves two gate delays or 6 nsec. The ROM access takes about seven gate delays or 21 nsec. The second XOR also takes 6 nsec. And loading the L and R registers takes about four gate delays (12 nsec). The total time for one iteration is thus 6 + 21 + 6 + 12 nsec = 45 nsec, and 16 iterations takes 720 nsec. Allowing 80 nsec for loading C, D, L, and R and for the check (see Figure 5) brings the total time to test one key to 800 nsec, or slightly less than the 1 μsec used in the basic argument. Even using the extremely conservative figure of 10 nsec/gate delay would result in a search time of less than 3 μsec per key, and, as noted earlier, the factor of three error is unimportant. Either the cost per solution is increased by this factor to $15,000 or we must wait about two and one half years for the decreasing cost of computation to erase the error. However, we should reemphasize that the 3 nsec/gate and 800 nsec/key are believed to be realistic with current technology.

Ciphertext-only attack

Up to now it has been assumed that a known plaintext-ciphertext pair was available for use in determining the key. While such known-plaintext attacks occur in practice and a good cryptosystem must be secure against them, a cryptanalyst would benefit greatly if he could also mount a successful attack without knowing any plaintext. Somewhat distressingly, under normal operating conditions the proposed NBS standard is vulnerable to such a ciphertext-only attack.

The cryptosystem is a FIPS (Federal Information Processing Standard) standard, binding on all applicable federal agencies. Because ASCII is also a FIPS standard, the plaintext data will often by represented as 8-bit ASCII characters. A plaintext block will consist of eight such characters, and since the last bit in each ASCII character is a parity bit, bits numbered 8, 16, 24, 32, 48, 56, and 64 in the plaintext block are the XOR of the preceding seven bits (e.g., bit 16 equals the XOR of bits nine through 15).

To take advantage of this fact, modify the chip so that one bit is added to the data given it after a reset command. This bit specifies whether a known-plaintext attack or a ciphertext-only attack is to be mounted. The known-plaintext attack proceeds as previously described, while in the ciphertext-only attack the chip is given two ciphertext blocks and its starting point in the key space. Upon entering the search mode the chip deciphers the first ciphertext block and checks if the eight parity positions in the computed plaintext block are valid parity bits. If the key tried was not the right key, there is one chance in $2^8 = 256$ of these parity bits appearing valid. (This assumes that the cryptodevice is, as it should be, a good pseudo-random number generator. We performed a simulation which supports the assertion that this probability is 1/256 when using the proposed standard.) By then doing the same test on the second ciphertext block the average false alarm rate is cut to one every $2^{16} \doteq 65{,}000$ attempts. This is low enough to prevent chip I/O from cutting into search rate. At one key/μsec a chip will

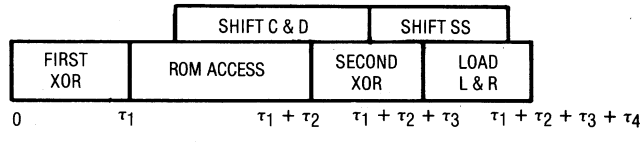


Figure 5. Inner loop timing diagram.

Table 2. Device estimates for major components.

COMPONENT	NUMBER OF DEVICES
56 BIT COUNTER	560
264 BITS OF SHIFT REGISTER (PARALLEL IN/OUT, DYNAMIC)	1584
CONTROL LOGIC (300 GATES)	900
2048 BIT ROM, INCLUDING DECODER	2148
144 XOR GATES	1008
64 INPUT OR GATE	192
TOTAL . . .	6392

have a tentative solution once every 65 msec, and the I/O bus on a 128-chip board will have to convey one such 56-bit solution every 0.5 msec, for an average load of 100 kbps. A 200-kbps I/O bus will easily handle this load. The board's control unit will have to be expanded slightly to include another search chip loaded with two different ciphertext blocks. All tentative solutions found on a board are double-checked by this device, cutting the false alarm rate by another factor of 65,000. The board will, therefore, convey a false alarm to the rack controller every 65,000 (0.5 msec) ≐ 30 sec, and a 128-board rack will receive false alarms at a rate of four per second. These are triple-checked by the rack with three additional search chips loaded with six different ciphertext blocks. Overall, a key must decipher 10 ciphertext blocks and produce proper parity before being passed to the minicomputer controller. Since there are 80 parity bits in these 10 blocks, the final false alarm rate is $2^{56}/2^{80} = 2^{-24} < 10^{-7}$ per search. In practice, a false solution is not to be expected.

Very little penalty is paid for modifying the machine to handle both known plaintext and ciphertext-only attacks. The number of gates added to the chip is about 200 for 64 XOR's, and the search rate is essentially unchanged since only one out of 256 keys has a doubled search time on the chip. The extra hardware in the board and rack control units is also minimal.

Variable key-size techniques

As indicated earlier, a standard with a variable key size is highly desirable. This section describes only two of the many ways to achieve this goal.

The first method is a variable-length key-expansion algorithm which appears to be at least as secure as the fixed-length expansion algorithm currently used in the standard. For illustrative purposes we have chosen 128 bits as the maximum length key, although the technique carries over just as well to 256 bits or other lengths.

First the key, followed by enough zeros to make a total of 128 bits, is used to drive a 128-bit linear feedback shift register.[17] The good psuedo-random properties of feedback shift register sequences ensure a good distribution within the final shift register contents even if a short key is used. After this initial set-up, we have 128 keying bits regardless of the original key size. The feedback connections are disabled and the shift register is used to shift its final contents cyclically eight positions after each of the 16 iterations of the basic encryption loop; 48 of the 128 stages are tapped to provide the 48 keying bits needed for each iteration. After the sixteenth iteration, the 128-bit shift register has returned to its initial position and is ready to encipher the next block. By reversing the direction of the shifts, the keying bits are obtained in reverse order for deciphering.

Another way to obtain a variable length key would be to use the currently proposed standard, but to encipher m times with m independent 56-bit keys. This would hopefully yield a $56m$-bit key, but additional analysis is required. For example, enciphering twice with two monoalphabetic ciphers is equivalent to enciphering only once with a third monoalphabetic cipher. If cipher one carries A to F and cipher two carries F to C, then the overall effect is to carry A to C. This is because monoalphabetic ciphers form a semi-group under composition. It is highly doubtful that the proposed standard possesses this property.

Even if there are 2^{56m} different transformations, the poor mixing of the m keys may allow shortcuts. For example, with $m = 2$, a "meet in the middle attack" allows cryptanalysis with a complexity of only 2^{56} in time and memory. Because

$$C = S_{K_2} S_{K_1}(P)$$

the intermediate variable

$$M = S_{K_1}(P) = S_{K_2}{}^{-1}(C)$$

can be defined. This last equation shows that if P is enciphered under all 2^{56} possible values for K_1 and C is deciphered under all 2^{56} possible values for K_2, then only those $K_1 - K_2$ pairs which have the same value of M are feasible solutions. A 128-bit block of text will yield a false solution rate of about 2^{-16} for the 112-bit quantity K_1, K_2.

The 2^{56} encipherments and decipherments required are not unreasonable with current technology. The memory requirements (2^{56} 128-bit words of memory), however, are several orders of magnitude above current capabilities. Advances in technology coupled with analysis to reduce the memory requirements will probably allow solution of doubly enciphered data within the next 10 or 20 years. We therefore recommend that, if multiple encipherment is used, m should be at least three.

While there are disadvantages to this technique when compared with the feedback shift-register key expander, it has the advantage of being compatible with the current standard. The disadvantages are decreased speed or increased hardware to allow m times as much computation, and that the user must do something special to get a high level of security. The latter problem might be overcome by making $m = 4$ the default option and requiring special action to obtain smaller values of m.

Discussion

It is now possible to see why the search machine should benefit fully from falling computation costs. Since the chip is compute bound, not I/O limited, it will receive full benefit from increases in gate speed. Also, as greater gate densities or larger chips become possible, many search devices can be put on one chip with a multiplexer to keep the pin count unchanged. The masking cost of such a superchip would hardly be more than that for a chip with a single search device since the mask would repeat the same basic search device many times.

The low cost per solution projected for the future ($50) is even possible today in many applications. If the key is composed of eight ASCII characters and these are chosen to be a sequence of letters (A through Z), then there are only $(26)^8 \doteq 2 \times 10^{11}$ keys to be searched. If the machine could be designed to search these keys first, it would find such a solution in much less than one second and at a cost of $0.014.

It may greatly complicate the search chip to effect this search pattern, but at least the seventh bit in each

character (which is a shift/control bit) can be held constant during the first phase of the search. Permuted choice one moves these bits (numbered 7, 15, 23, 31, 39, 47, 55, and 63) together, facilitating such a search mode. The machine would take less than six minutes to exhaust these 2^{48} keys. The corresponding average cost per solution is cut from $5000 to $20.

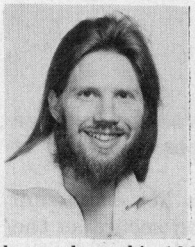
In summary, we believe we have made a convincing argument concerning the insecurity and planned obsolescence inherent in the proposed cryptostandard. We have also indicated cost-effective ways to avoid the problem. A 128-bit or larger key is needed to preclude exhaustive search and to allow a margin of safety against shortcuts to exhaustive search. We hope that those readers who will be most affected by this standard will let their views be known to NBS. ■

Acknowledgment

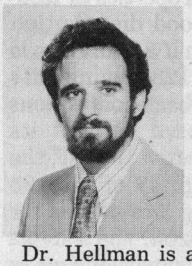
The authors wish to thank the many people who provided information and advice. In particular Dr. Roger Melen of Stanford's Integrated Circuits Lab and Prof. Forest Baskett III of Stanford's Computer Science Department provided invaluable advice in their areas of expertise. Bruce Marion performed the simulation; several IC industry experts, including one of the reviewers, provided very helpful suggestions.

References

1. *Federal Register,* March 17, 1975, Vol. 40, No. 52.
2. *Federal Register,* August 1, 1975, Vol. 40, No. 149.
3. W. Diffie, "Preliminary Remarks on the National Bureau of Standards Proposed Standard Encryption Algorithm for Computer Data Protection," sent to NBS, May 22, 1975.
4. M. Hellman, letter to Dennis Branstad at NBS, dated October 22, 1975.
5. E. K. Yasaki, "Encryption Algorithm: Key Size is the Thing," *Datamation,* March 1976, pp. 164-166.
6. W. Diffie and M. Hellman, "New Directions in Cryptography," *IEEE Trans. on Information Theory,* Vol. IT-22, November 1976, pp. 644-654.
7. S. Jeffery (NBS), letter to M. Hellman dated January 6, 1976.
8. L. Brillouin, *Science and Information Theory,* Academic Press, London, second edition, 1962.
9. R. W. Keyes, "Physical Limits in Digital Electronics," *Proceedings of the IEEE,* Vol. 63, May 1975, pp. 740-767.
10. National Bureau of Standards, "Report of the 1976 Workshop on Estimation of Significant Advances in Computer Technology," August 30-31, 1976.
11. M. Hellman,, R. Merkle, R. Schroeppel, L. Washington, W. Diffie, S. Pohlig, and P. Schweitzer, "Results of an Initial Attempt to Cryptanalyze the NBS Data Encryption Standard," Information Systems Laboratory Report, Stanford University, November 10, 1976. (Available from NTIS.)
12. R. Morris, N. J. A. Sloane, and A. D. Wyner, "Assessment of the National Bureau of Standards Proposed Federal Data Encryption Standard," Bell Telephone Laboratories memorandum, December 1976; also to appear *Cryptologia.*
13. D. Grubb, NBS memo to Dr. Dennis Branstad. Subject: Stanford University letter on length of key needed for NBS encryption algorithm. Dated November 4, 1975.
14. R. Davis, NBS letter to the Hon. Charles A. Mosher (U.S. House of Representatives) dated March 30, 1975.
15. F. Faggin, "Microprocessor Technology: Yesterday, Today and Tomorrow," *EDN,* November 20, 1975.
16. F. Faggin, "The Role of Technology in Microcomputer Design and Evolution," *IEEE Trans. on Circuits and Systems,* Vol. 7, No. 5, February 1975, pp. 4-13.
17. S. Golomb, *Shift Register Sequences,* Holden Day, San Francisco, 1967.
18. C. Shannon, "The Communication Theory of Secrecy Systems," *BSTJ,* Vol. 28, pp. 656-715, 1949.
19. H. Hellman, "An Extension of the Shannon Theory Approach to Cryptography," *IEEE Trans. on Information Theory,* Vol. IT-23, May 1977, pp. 289-294.
20. A. Carleial and M. Hellman, "On Wyner's Wire-tap Channel," *IEEE Trans. on Information Theory,* Vol. IT-23, May 1977, pp. 387-390.



Whitfield Diffie is a graduate student doing research in cryptography in the Department of Electrical Engineering at Stanford University. He previously worked for the Mitre Corporation and at the Artificial Intelligence Laboratories of both MIT and Stanford, doing research in symbolic mathematical manipulation and the theory of program correctness. He holds a bachelor of science degree in mathematics from MIT, from which he graduated in 1965.



Martin E. Hellman, an associate professor of electrical engineering at Stanford University, is doing research in cryptography and communication theory. He was an assistant professor at MIT from 1969-1971 and on the staff of IBM's Watson Research Center during 1968-1969. He received the BE degree from NYU in 1966 and the MS and PhD degrees from Stanford in 1967 and 1969, all in electrical engineering.

Dr. Hellman is a member of the IEEE Information Theory Group's Board of Governors and was an associate editor of the *Transactions on Communications.* He is the author of over 25 articles.