To appear in the Proceedings of the 1982 Symposium on Security and Privacy (IEEE)

# SHARED DATABASE ACCESS USING COMPOSED ENCRYPTION FUNCTIONS

Neal R. Wagner

Drexel University
Mathematical Sciences Department
Philadelphia, Pennsylvania 19104

## Abstract

This article presents a two-stage encryption method for sharing access to a database where no single agency or device can ever encrypt or decrypt the data directly. Thus an attack by an opponent would have to succeed at two separate points. The main tool needed is a secure cryptosystem closed under composition: encrypting and re-encrypting using two successive keys is equivalent to a single encryption using some third key. An example cryptosystem satisfying this condition is exponentiation modulo a fixed prime.

## 1. Introduction

Suppose we wish to allow n users to share access to a single encrypted file F. Straightforward systems might employ one of two basic types of approaches:

(a) give each user the key $K_F$ to the file, or

(b) let each user employ a separate key $K_i$ ($1 \le i \le n$) and allow some trusted central authority, which we call the Data Distributor (DD), to give parts of F to the user by decrypting with $K_F$ and encrypting with $K_i$.

Approach (a) has the disadvantage of wide distribution of the file key $K_F$. A breach in the security of any single user compromises the security of the entire file. Approach (b), the natural choice for high security, has many variations and extensions. There are methods requiring relatively little space which allow the DD to derive any number of file keys for different coalitions of users [Den81]. There are proposals which avoid storing the various $K_i$ or $K_F$ in the clear [Dif76], [Eva74] and proposals requiring several users to give their keys to the DD for combination into the file key $K_F$ [Bla79], [Sha79]. However, in each case the DD employs the file key directly for decryption and produces parts of the file as cleartext. The file key and the cleartext exist in the hands of the DD, at least transiently. At best we could limit the cleartext to a brief existence in a sealed piece of hardware before it was re-encrypted for a given user. But a better solution, which we propose here, would prevent the cleartext from existing in any form until final decryption by an individual user.

Normally it is the goal of cryptography to reduce the possible weak points of a system to a single cryptographic key. However, here we are trying to extend security beyond this -- to a system which requires an opponent to compromise the system at two separate points.

This paper presents a scheme involving the factoring of the key $K_F$ any number of ways as the composition of a user key $K_i$ and a complementary key $L_i$ held by the DD. These factored keys are a generalization to several users of a simple two-stage encryption scheme for one user. (Notice that we are factoring keys, not numbers. The factoring described here has no direct connection with the factoring of integers that is part of the security of the RSA cryptosystem [Riv78].)

For simplicity we will use the same symbol for a key and for the encryption function which uses the key. Thus the function which encrypts using key K is denoted by {-}K or just by K. Assuming the messagespace is the same as the cipherspace, double encrytion by successive keys $K_1$ and $K_2$ will be denoted by { $\{-\}K_1\}K_2$ or just by $K_1 \circ K_2$. Notice that $K_1 \circ K_2$ means first $K_1$ and then $K_2$, since we are writing encryption functions on the right. For an encryption function K, denote by $K^{-1}$ the corresponding decryption function.

One simple approach to writing $K_F$ as $K_i \circ L_i$ for any number of i would choose $K_i$ randomly and set $L_i = K_i^{-1} \circ K_F$. Alternatively we could choose $L_i$ randomly and set $K_i = K_F \circ L_i^{-1}$. However, these approaches are only satisfactory if the composition can be rewritten as a single unified operation not explicitly involving $K_F$. Unless so unified, the first form would violate our desire never to have the cleartext exist until the final decryption. The second form, again unless so unified, would place $K_F$ into the hands of the user.

## 2. Keys closed under composition

For simplicity we make a stronger assumption than just that keys can be factored. Assume we are working with a cryptosystem satisfying the following properties:

(i) Closure under composition. The messagespace is the same as the cipherspace, and for any keys $K_1$ and $K_2$, encryption under $K_1$ followed by further encryption under $K_2$ is equivalent to encryption under $K_3$ for some feasibly computable key $K_3$. In other words,

$$K_1 \circ K_2 = K_3, \text{ for some } K_3.$$

(ii) Security (intuitive). Given a key $K_3$, either it must be an intractable problem to discover solutions to the equation $K_1 \circ K_2 = K_3$ in the unknown keys $K_1$ and $K_2$, or there must be so many solutions that it is intractable to discover a specific one. A knowledge of plaintext-ciphertext pairs resulting from the use of any of the unknown keys should not help in solving. It must further be intractable to solve more complex systems of equations in unknown keys as long as the corresponding modular integer equations are not uniquely solvable (replacing "$\circ$" by multiplication and key inverse by ordinary modular arithmetic inverse).

This property implicitly assumes an implementation like the one presented in the next section. A different implementation might allow a weaker security assumption. For our applications in this section we also need:

(iii) Symmetry. For any key K, it is feasible to calculate the inverse key $K^{-1}$ which decrypts the encryption performed by K.

An encryption system based on exponentiation modulo a fixed prime satisfies these properties, as will be described in the next section. For now, we want to describe the use of a system satisfying (i), (ii), and (iii).

We wish to give procedures to build up keys $K_1, K_2, \ldots, K_n$ separately in the hands of n users and complementary keys $L_1, L_2, \ldots, L_n$ in the hands of the DD. Each of the compositions $K_i \circ L_i$ $(1 \le i \le n)$ will give the same encryption function. (See Figure 1.)

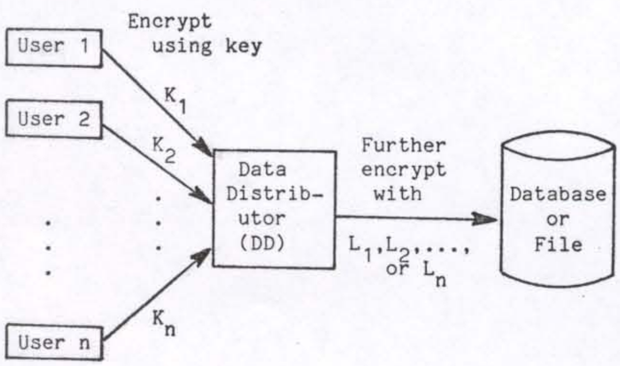


Figure 1. Encryption into the database. (Each double encryption $K_i \circ L_i$ is the same.)

The construction is carried out in such a way that at no time is any single individual or device in a position to encrypt data for the file F or to decrypt data contained in F. The construction for the first user is different from that for any subsequent user. The first user simply cooperates with the DD to produce the desired keys. Each subsequent user must cooperate both with the DD and with a "sponsoring" user already enrolled in the system.

Key Distribution -- User 1
(Requires cooperation between User 1 and the DD.)
User 1 chooses random keys X and $K_1$. User 1 then forms

$$Z_1 = K_1^{-1} \circ X$$

and sends $Z_1$ to the DD. The DD chooses a random key Y and calculates

$$L_1 = Z_1 \circ Y.$$

User 1 encrypts using $K_1$ and the DD further encrypts using $L_1$. The composition of these is

$$K_1 \circ L_1 = K_1 \circ Z_1 \circ Y = K_1 \circ K_1^{-1} \circ X \circ Y = X \circ Y.$$

Only User 1 knows X and only the DD knows Y, but neither knows both, and neither can calculate both efficiently (by property (ii)).

Key Distribution -- User n $(n \ge 2)$
(Requires cooperation between User n, the DD, and some sponsoring User i, for $i < n$.)
User n chooses a random key $K_n$ and sends $K_n$ to a sponsor, say User i, who calculates

$$Z_n = K_n^{-1} \circ K_i.$$

User i then sends $Z_n$ to the DD, who calculates

$$L_n = Z_n \circ L_i.$$

Here a simple induction argument shows that

$$K_n \circ L_n = K_n \circ Z_n \circ L_i = K_n \circ K_n^{-1} \circ K_i \circ L_i =$$

$$K_i \circ L_i = X \circ Y,$$

so that again

$$K_n \circ L_n = X \circ Y.$$

The sponsoring user will know User n's secret key $K_n$, but we will shortly see how to eliminate this problem.

Security of this system again follows from property (ii). For example, if i = 1, the DD has in hand both $Z_1$ and $Z_n$, where

$$Z_1 = K_1^{-1} \circ X, \text{ and}$$

$$Z_n = K_n^{-1} \circ K_1.$$

These are two equations involving three unknown keys, and the corresponding integer equations using multiplication for "$\circ$" and modular arithmetic are not uniquely solvable for any of the unknowns. Hence by (ii) the DD should not be able to deduce $K_1$, $K_n$ or X.

If the secret key of any user is compromised, there is a simple way to change keys without a sponsoring user.

Change of User Key -- User i
User i chooses a random secret key V and sends it to the DD, who changes $L_i$ to

$$L_i' = V^{-1} \circ L_i.$$

User i changes his secret key to

$$K_i' = K_i \circ V.$$

The remaining $K_j$ and $L_j$ for j not equal i stay the same. For User i, encryption for the file F now uses the composition

$$K_i' \circ L_i' = K_i \circ V \circ V^{-1} \circ L_i = K_i \circ L_i = X \circ Y,$$

which is the same as before.

It is now clear that there is a better way to do the key distribution construction for all but the first user, combining the old construction with a key change, so that the sponsoring user no longer learns the secret key. The key distribution for User 1 remains the same. (See Figure 2.)

Improved Key Distribution -- User n ($n \geq 2$)
(Requires cooperation between User n, the DD, and some sponsoring User i, for i < n.)
User n chooses random keys U and V and calculates

$$K_n = U \circ V.$$

User n sends U to a sponsor, say User i, who calculates

$$Z_n = U^{-1} \circ K_i.$$

User i then sends $Z_n$ to the DD. User n sends V directly to the DD, who calculates

$$L_n = V^{-1} \circ Z_n \circ L_i.$$

Here we have

$$K_n \circ L_n = U \circ V \circ V^{-1} \circ U^{-1} \circ K_i \circ L_i = K_i \circ L_i = X \circ Y,$$

which can be shown by an inductive argument. Security follows from assumption (ii) as before.

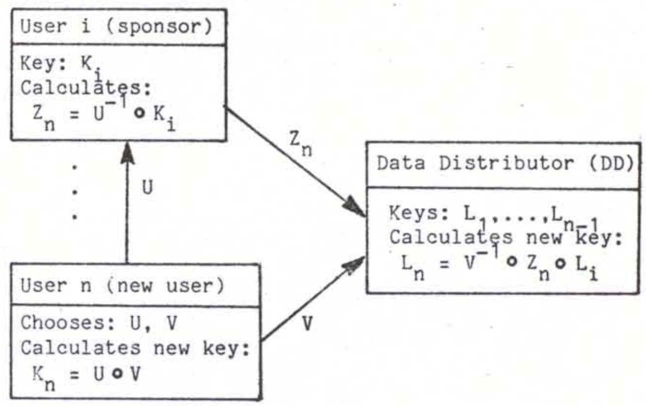


Figure 2. Key distribution for User n ($n \geq 2$), with cooperation from the DD and User i.

If the DD's collection of $L_i$'s is ever compromised, the $L_i$'s and the entire encrypted file F can be restructured as follows.

Restructuring the $L_i$ and the file F
The DD chooses a random key W. Each $L_i$ is changed to

$$L_i' = L_i \circ W,$$

and each ciphertext C is changed to

$$C' = \{C\}W.$$

The resulting new $L_i'$ and the old $K_i$ work together as before for encryption into or decryption out of the new version of the file.

## 3. Implementation by exponentiation modulo a prime

In general we do not expect a cryptosystem to be closed under composition. For example the Data Encryption Standard (DES) [Des77], with its non-linear S-boxes, should seldom exhibit this closure. The DES has $2^{56}$ different keys, and so it has at most $2^{56}$ different encryption functions. Since the DES encrypts 64-bit blocks (when used in block mode), there are $(2^{64})!$ different possible encryption functions [Kon81, p. 64] -- a very large number indeed. The chances that a pair of DES keys, when composed, might yield another DES key should be very small, though the author does not know whether this ever occurs.

Some cryptosystems are closed under composition. One example is the Hill cryptosystem [Kon81], which multiplies a non-singular matrix of integers times the message as a vector of integers. However, use of the Hill system is not recommended, primarily because of its linearity.

Fortunately, one common cryptosystem of apparent high security is closed under composition: exponentiation modulo a prime. If P is a fixed "large" prime and K satisfies $0 < K < P-1$ and is relatively prime to P-1, then

$$\{M\}K = M^K \bmod P$$

is an encryption function satisfying (i) and (iii) of Section 2. This encryption function is also thought to be secure [Adl79], [Poh78], but proofs of security may never be in hand [Bla80]. (This method has the further interesting property of commutativity:

$$K_1 \circ K_2 = K_2 \circ K_1, \text{ for all } K_1 \text{ and } K_2.$$

Commutativity is not needed for our applications, though it plays an essential role in the public key distribution method of [Dif76] and in the "mental poker" of [Sha78].)

To see that (i) holds, choose $K_1$ and $K_2$ both less than P-1 and relatively prime to P-1. Then for any M,

$$\{\ \{M\}K_1\}K_2 = (M^{K_1} \bmod P)^{K_2} \bmod P =$$

$$M^{K_1 \cdot K_2} \bmod P = M^{K_3} \bmod P,$$

where $K_3 = (K_1 \cdot K_2) \bmod (P-1)$. (See [Poh78].)

For (iii), given K, we define $K^{-1}$ as the solution to the equation

$$(K \cdot K^{-1}) \bmod (P-1) = 1,$$

and there are efficient methods to calculate such a $K^{-1}$ [Knu81, Exer. 4.5.2.15].

As discussed in [Poh78], in order that exponentiation modulo P be secure, P must be chosen so that P-1 has a large prime factor. Since P is fixed for the whole system, one convenient possibility is to choose a prime of the form

$$P = 2^k \cdot Q + 1,$$

where Q is also a large prime and $2^k$ is small. For a given Q, such a prime P might not exist [Bai81], but since we can try any number of different Q's, a prime like P is readily constructed, even with $k = 1$ [Poh78]. Recent work [Hel80] suggests that a P of size 80 decimal digits represents the lower limit for security right now.

A prime of the form $2^k \cdot Q + 1$ (with $2^k$ small) makes it easy to find integers to serve as keys, since we need only choose a random odd number less than $P - 1$. (The chance that such a number might have Q as a divisor is so small it can be neglected.)

It is also important that the message being encrypted not have too small an order. For example, the "message" $P - 1$ yields 1 when raised to any power mod P. However, only $2^k$ messages have order $\leq 2^k$, while the remaining messages have order $\geq Q$. Assuming Q is very large and $2^k$ is small, there will be a vanishingly small chance of encountering such a message.

4. Use of the RSA Cryptosystem

Instead of exponentiation modulo a prime, suppose we consider exponentiation modulo N, where N is the product of two primes P and Q: the so-called RSA cryptosystem [Riv78]. If P and Q are known to all participants, then the resulting system satisfies all the properties stated in Section 2 and can be used exactly as exemplified in Section 3. In fact [Hel80] recommends routine use of RSA in place of exponentiation mod P, and this would be particularly reasonable if RSA is implemented in special hardware [Riv80].

However, the possibility of keeping the primes P and Q secret allows us to create a system in which the keys themselves enforce read-only, write-only or read-write access to a shared file, while maintaining the extra two-step process for encryption or decryption. In this system the key distribution must be carried out by a trusted Central Authority (CA). But once the CA is finished, we are again in the position that no single individual can encrypt or decrypt alone.

User i employs

$$C = M^{E_i} \bmod N, \text{ and } M = C^{D_i} \bmod N$$

for encryption and decryption, where

$$(D_i \cdot E_i) \bmod (P-1)(Q-1) = 1.$$

The CA gives the DD keys $F_i$ and $G_i$ for each user satisfying

$$E_i \circ G_i = E_F \text{ and } F_i \circ D_i = D_F,$$

where $E_F$ is the encryption key for the file F and $D_F$ is the decryption key.

In order to give read-only access, the CA provides User i with only $D_i$. For write-only access, User i gets only $E_i$. For read-write access, User i gets both $D_i$ and $E_i$, and thus can deduce the secret primes P and Q [Riv78]. If any single user has read-write access, then the DD will need to know the secret primes, so we might as well assume the DD has both the keys $F_i$ and $G_i$ for each User i. Because a user with read-write access can deduce the P and Q, such a user can grant this access to other users with read-only or write-only access. However, two users, one with read-only and one with write-only access, cannot combine their information and obtain read-write access individually. (A similar scheme for using the RSA cryptosystem to limit privileges was proposed in [Mu82b] and has surely been considered by others. We mention it here because it fits in with our shared database method.)

5. Coalitions of users

The basic system discussed in Sections 2 and 3 extends to a system requiring more than one user in addition to the DD in order to access the database. For this application, we would usually want the commutativity provided by the example system in Section 3.

For example, suppose we wanted a <u>pair</u> of users and the DD to be required for encryption or decryption. Each user has a key $K_i$ as before. The users in pairs could allow the DD to calculate

$$L_{ij} = K_i^{-1} \circ K_j^{-1} \circ X \circ Y, \text{ for each } i < j$$

as in Section 2, or for a large number of users this could be done by an authentication server. As before, there are no unique solutions for the $K_i$, so this system is secure by (ii) of Section 2. With n users, this requires the DD to store n(n-1)/2 keys, which does not compare favorably to the methods of [Bla79] and [Sha79]. However, in practice we might only need to set this up for a much more limited number of pairs.

In order to encrypt for the database, Users i and j encrypt successively in either order with $K_i$ and $K_j$, and then the DD further encrypts with $L_{ij}$. (If we did not wish to use the commutativity, we could specify the order.) Decryption is the reverse process, and requires that the two users who plan to decrypt identify themselves to the DD, so that the proper $L_{ij}$ will be employed.

More complicated arrangements are possible: for example for any fixed $k \geq 1$, we can set things up so that any set of k users and only such sets can encrypt or decrypt data with the help of the DD. If n is large and k is not very small, then the number of keys the DD must store, n!/(k!(n-k)!), will be unreasonably large. Again, however, the system would be workable for certain specific sets of k users. We can also use any mutually disjoint collections of subsets of users, and these two ideas can be combined. For example, we might have one class of users who could access the database without help from any other user (but of course with help from the DD), and another separate class that could only access the same database in pairs.

There exist collections of subsets of users for which the system cannot be set up without allowing the DD to deduce secret user keys. (For example, <u>all</u> subsets.)

## 6. Sample calculations

We present some realistic calculations illustrating the basic system described in Section 2, using the cryptosystem of Section 3: exponentiation modulo a fixed prime P. (See Figure 3 at the end of this section.)

In what follows, the "primes" are actually probabilistic pseudoprimes [Knu81]. From any practical point of view, these can be made equivalent to true primes. We generated several random 100-digit pseudoprimes Q until we were able to find a pseudoprime $P = 2^k \cdot Q + 1$ with small k. After a dozen tries, we ended up with k = 11 and a pseudoprime

P = 4104191113 1811520776 5247657866 2841971894
9811785304 0231811111 4553084369 8367084374
4782814659 1343469324 289.

For this simple example we will restrict to just two users: User 1 and User 2. Suppose User 1 chooses random keys

$K_1$ = 3189307680 4101548203 6711515891 8486671929
2141203460 8408113403 1173427491 5456020386
4061159466 4653981435 559,

X = 9752516527 0482736970 6538629626 0081342682
0705958629 9393680748 4298379320 5709795583
0858469534 4108322353 43.

User 1 calculates $Z_1 = K_1^{-1} \circ X$ and sends $Z_1$ to the DD:

$Z_1$ = 1308210187 1246966272 9608945039 8324234547
5208891298 8311165796 2847265085 8995636133
9753669102 2097694597 913.

Suppose the DD chooses a random key

Y = 3684289911 5552527041 6932405634 9721579710
5970330874 9295135748 7591067915 4974349211
1512636547 4627307565 413.

The DD then calculates and saves $L_1 = Z_1 \circ Y$:

$L_1$ = 2562055169 4235177696 2212519109 7053819032
9038985633 2080101299 8288749332 3983625144
7840557587 1650774534 621.

Suppose we take the following as the cleartext "message" M to be included into the database:

M = THISOISOAO CLEARTEXTO BLOCKOREAD YOFOROENCR
YPTION,

where M is represented base 36, with "A" = 10, "B" = 11, ..., "Z" = 35. User 1, with help from the DD, can get M into the database in two stages. First User 1 will encrypt M using $K_1$ to produce an intermediate ciphertext C' equal to M raised to the power $K_1$, mod P,

C' = 1112186002 9575147256 4888632903 9739861108
2613473437 3621467637 3652869438 4443320921
8393275545 7669072677 167.

Then the DD uses the key $L_1$ to further encrypt this to a ciphertext C which is actually put into the database. C is equal to C' raised to the power $L_1$, mod P,

C = 1221699207 3349776455 7436071398 0962619116
8321063589 1242297918 0440318055 4854236800
5237946815 3043205245 745.

Now we add User 2 to the system. User 2 chooses random keys

U = 8289104389 9452105393 8791512170 5163352920
8095118286 5375965120 2880080089 5359889830
4239348025 1746981153 21,

$$
\begin{aligned}
V = {} & 1387044733\ 0002803666\ 2482850578\ 1005699175 \\
& 7209474514\ 9449684327\ 5210194448\ 1087851239 \\
& 8824581189\ 5441292797\ 133.
\end{aligned}
$$

User 2 uses the composition of these as his secret key $K_2 = U \circ V$:

$$
\begin{aligned}
K_2 = {} & 2853516774\ 5662431950\ 9947115928\ 8965034469 \\
& 8271806608\ 8269204793\ 2053824663\ 7655278549 \\
& 1195949479\ 3542338364\ 261.
\end{aligned}
$$

User 2 sends U to User 1, who calculates $Z_2 = U^{-1} \circ K_1$, and User 1 sends $Z_2$ to the DD:

$$
\begin{aligned}
Z_2 = {} & 3750633637\ 8427332383\ 7041477046\ 5466671130 \\
& 2883746632\ 0818289728\ 5914372261\ 1481163613 \\
& 2508610980\ 8601804887\ 199.
\end{aligned}
$$

User 2 sends V directly to the DD, who calculates $L_2 = V^{-1} \circ Z_2 \circ L_1$:

$$
\begin{aligned}
L_2 = {} & 2553576281\ 2683297383\ 0368892557\ 9084420028 \\
& 1247674239\ 1822629108\ 5413658903\ 5788352550 \\
& 5505130724\ 4143770709\ 327.
\end{aligned}
$$

User 2 can now retrieve M by first asking the DD to decrypt C with $L_2^{-1}$ to get the intermediate ciphertext C'' equal to C raised to the power $L_2^{-1}$, mod P,

$$
\begin{aligned}
C'' = {} & 5005525650\ 8070937687\ 9161001983\ 4487173040 \\
& 3375845019\ 3808992274\ 6810547965\ 9909259666 \\
& 4577550780\ 9788312230\ 79.
\end{aligned}
$$

Finally User 2 further decrypts C'' with $K_2^{-1}$ by raising C'' to the power $K_2^{-1}$, mod P. This produces for User 2 the original message M that User 1 added to the database, written below in base 10:

$$
\begin{aligned}
M = {} & 3185949415\ 2837306306\ 5074412804\ 9610839928 \\
& 4792451507\ 9356293649\ 2230558486\ 63.
\end{aligned}
$$

Notice that the DD cannot obtain M. The ciphertext C in the database could be obtained by encrypting with $X \circ Y$, though no one is in a position to carry out this calculation.

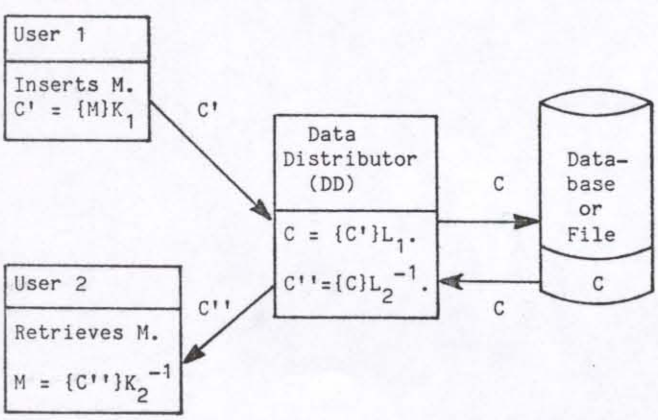


Figure 3. Insertion of M into the database by User 1 and retrieval by User 2.

## 7. Conclusions

We have presented a shared database which uses a two-stage encryption to force an opponent to attack at two separate points, assuming the underlying cryptosystem is secure. This proposed system would be especially reasonable for applications requiring the very highest degree of security. In that case other more conventional security measures should be simultaneously employed. In particular, some form of authentication of users would be required, so an authentication server [Nee78] or at least some relatively simple central authority [Mu82a] would be needed.

The proposed system requires separate user keys for separate files. If this key storage becomes a burden, the user could easily employ pseudo-random keys, generated by applying a one-way hash function to the file name and a single secret user key.

We have suggested exponentiation modulo a fixed prime for use in a specific implementation of the system. Without special hardware this cryptosystem is rather computation intensive, but the same can be said about the Data Encryption Standard. One special chip will easily keep up with a phone line [Riv80], and faster encryption rates could be provided by more elaborate hardware.

The best use of this system might be for an application with many users communicating relatively little data to a relatively large database. In this case an opponent who compromises one user cannot produce large amounts of cleartext without attracting attention, while the database itself is secure unless an opponent can compromise both a user and the Data Distributor (as well as gain physical access to the database).


## Acknowledgment

This work was carried out with partial support from Siemens Corporation and using the computer of Drexel University's Center for Scientific Computation. Dr. C. Mueller-Schloer of Siemens suggested the idea of factored keys and provided many helpful comments. Dr. E. Leiss also read the manuscript carefully and suggested a number of improvements.


## References


[Adl79] L. Adleman, "A subexponential algorithm for the discrete logarithm problem with applications to cryptography," Proceedings of the Twentieth IEEE Symposium on Foundations of Computer Science, Oct. 1979, pp. 55-60.

[Bai81] R. Baillie, G. Cormack, and H. C. Williams, "The problem of Sierpinski concerning $k \cdot 2^n + 1$," Mathematics of Computation 37, 155 (July 1981), pp. 229-231.

[Bla79] B. R. Blakely, "Safeguarding cryptographic keys," AFIPS Conference Proceedings 48, 1979 National Computer Conference, pp. 313-317.

[Bla80] B. R. Blakely, "One-time pads are key safeguarding schemes, not cryptosystems. Fast key safeguarding schemes (threshold schemes) exist," Proceedings of the 1980 Symposium on Security and Privacy, IEEE Computer Society, pp. 108-113.

[Den81] D. E. Denning, and F. B. Schneider, "Master keys for group sharing," Information Processing Letters 12, 1 (13 Feb. 1981), pp. 23-25.

[Des77] Data Encryption Standard, Federal Information Processing Standard (FIPS) Publication 46, National Bureau of Standards, U. S. Department of Commerce, Washington, DC (Jan. 1977).

[Dif76] W. Diffie, and M. E. Hellman, "New directions in cryptography," IEEE Transactions on Information Theory IT-22, 6 (Nov. 1976), pp. 644-654.

[Eva74] A. Evans, W. Kantrowitz, and E. Weiss, "A user authentication system not requiring secrecy in the computer," Communications of the ACM 17, 8 (Aug. 1974), pp. 437-442.

[Hel80] M. E. Hellman, "On the difficulty of factoring logarithms over $GF(q^m)$," Proceedings of the 1980 Symposium on Security and Privacy, IEEE Computer Society, p. 83.

[Knu81] D. Knuth, The Art of Computer Programming, Vol II: Semi-numerical Algorithms, Second Edition, Addison-Wesley, 1981.

[Kon81] A. G. Konheim, Cryptography. A Primer, Wiley, 1981.

[Mu82a] C. Mueller-Schloer, and N. R. Wagner, "The implementation of a cryptography-based secure office system," accepted for the 1982 National Computer Conference.

[Mu82b] C. Mueller-Schloer, "Cryptographic protection of personal data cards," submitted to the Seventh International Conference on Computer Communication.

[Nee78] R. Needham, and M. Schroeder, "Security and authentication in large networks of computers," Communications of the ACM 21, 12 (Dec. 1978), pp. 993-999.

[Poh78] S. C. Pohlig, and M. E. Hellman, "An improved algorithm for computing logarithms over GF(p) and its cryptographic significance," IEEE Transactions on Information Theory IT-24,1 (Jan. 1978), pp. 106-110.

[Riv80] R. L. Rivest, "A description of a single-chip implementation of the RSA cipher," Lambda 1, 3 (1980), pp. 14-18.

[Riv78] R. L. Rivest, A. Shamir, and L. Adleman, "A method for obtaining digital signatures and public-key cryptosystems," Communications of the ACM 21, 2 (Feb. 1978), pp. 120-126.

[Sha79] A. Shamir, "How to share a secret," Communications of the ACM 22, 11 (Nov. 1979), pp. 612-613.

[Sha78] A. Shamir, R. L. Rivest, and L. M. Adleman, "Mental Poker," Technical Report No. TM-125, MIT Laboratory for Computer Science (Nov. 1979).