Data Encryption

Encryption refers to the coding of information in order to keep it secret. Encryption is accomplished by transforming the string of characters comprising the information to produce a new string that is a coded form of the information. This is called a **cryptogram** or **ciphertext** and may be safely stored or transmitted. At a later time it can be deciphered by reversing the encrypting process to recover the original information, which is called **plaintext**.

Data encryption has been used to send secret military and political messages from the days of Julius Caesar to the present. Recent applications include the Washington-Moscow hotline, electronic funds transfer, electronic mail, database security, and many other situations in which the transmission of secret data is crucial. Less profound applications have included Captain Midnight secret decoder rings that could be obtained in the 1950s for twenty-five cents and two Ovaltine labels, puzzles appearing in the daily newspaper, and a number of other frivolous applications. In this section we describe some encryption schemes ranging from the Caesar cipher scheme of the first century B.C. to the Data Encryption Standard and the public key encryption schemes of the 20th century.

The simplest encryption schemes are based on the string operation of **substitution,** in which the plaintext string is traversed and each character is replaced by some other character according to a fixed rule. For example, the **Caesar cipher** scheme consists of replacing each letter by the letter that appears *k* positions later in the alphabet for some integer *k*. (The alphabet is thought of as being arranged in a circle, with A following Z.) In the original Caesar cipher, *k* was 3, so that each occurrence of A in the plaintext was replaced by D, each B by E, . . ., each Y by B, and each Z by C. For example, using the character set

A B C D E F G H I J K L M N O P Q R S T U V W X Y Z

we would encrypt the string “IDESOFMARCH” as follows:

To decode the message, the receiver uses the same **key** *k* and recovers the plaintext by applying the inverse transformation, that is, by traversing the ciphertext string and replacing each character by the character *k* positions earlier in the alphabet. This is obviously not a very secure scheme, since it is possible to “break the code” by simply trying the 26 possible values for the key *k*.

An improved substitution operation is to use a **keyword** to specify several different displacements of letters rather than the single offset *k* of the Caesar cipher. In this **Vignère cipher** scheme, the same keyword is added character by character to the plaintext string, where each character is represented by its position in the character set and addition is carried out modulo 26. For example, suppose the character set and positions of characters are given by

Position	0	1	2	3	4	5	6	7	8	9	10	11	12
Character	A	B	C	D	E	F	G	H	I	J	K	L	M

13	14	15	16	17	18	19	20	21	22	23	24	25
N	O	P	Q	R	S	T	U	V	W	X	Y	Z

and that the keyword is DAGGER. The plaintext IDESOFMARCH is then encrypted as follows:

Plaintext:	I	D	E	S	O	F	M	A	R	C	H
Repeated key word:	D	A	G	G	E	R	D	A	G	G	E
Ciphertext:	L	D	K	Y	S	W	P	A	X	I	L

Again, the receiver must know the key and recovers the plaintext by subtracting the characters in this keyword from those in the ciphertext.

A different substitution operation is to use a ***substitution table,*** for example:

Original character:	A	B	C	D	E	F	G	H	I	J	K	L	M
Substitute character:	Q	W	E	R	T	Y	U	I	O	P	A	S	D

N	O	P	Q	R	S	T	U	V	W	X	Y	Z
F	G	H	J	K	L	Z	X	C	V	B	N	M

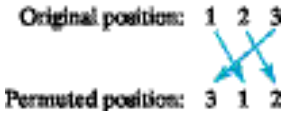
The string IDESOFMARCH would then be encoded as follows:

Plaintext:	I	D	E	S	O	F	M	A	R	C	H
Ciphertext:	O	R	T	L	G	Y	D	Q	K	E	I

To decode the ciphertext string, the receiver must again know the key, that is, the substitution table.

Since there are 26! (approximately 10^{28}) possible substitution tables, this scheme is considerably more secure than the simple Caesar cipher scheme. Experienced cryptographers can easily break the code, however, by analyzing frequency counts of certain letters and combinations of letters.

Another basic string operation in some encryption schemes is **permutation**, in which the characters in the plaintext or in blocks of the plaintext are rearranged. For example, we might divide the plaintext string into blocks (substrings) of size 3 and permute the characters in each block as follows:



Thus the message IDESOFMARCH is encrypted (after the addition of a randomly selected character X so that the string length is a multiple of the block length)

To decode the ciphertext string, the receiver must know the key permutation and its inverse

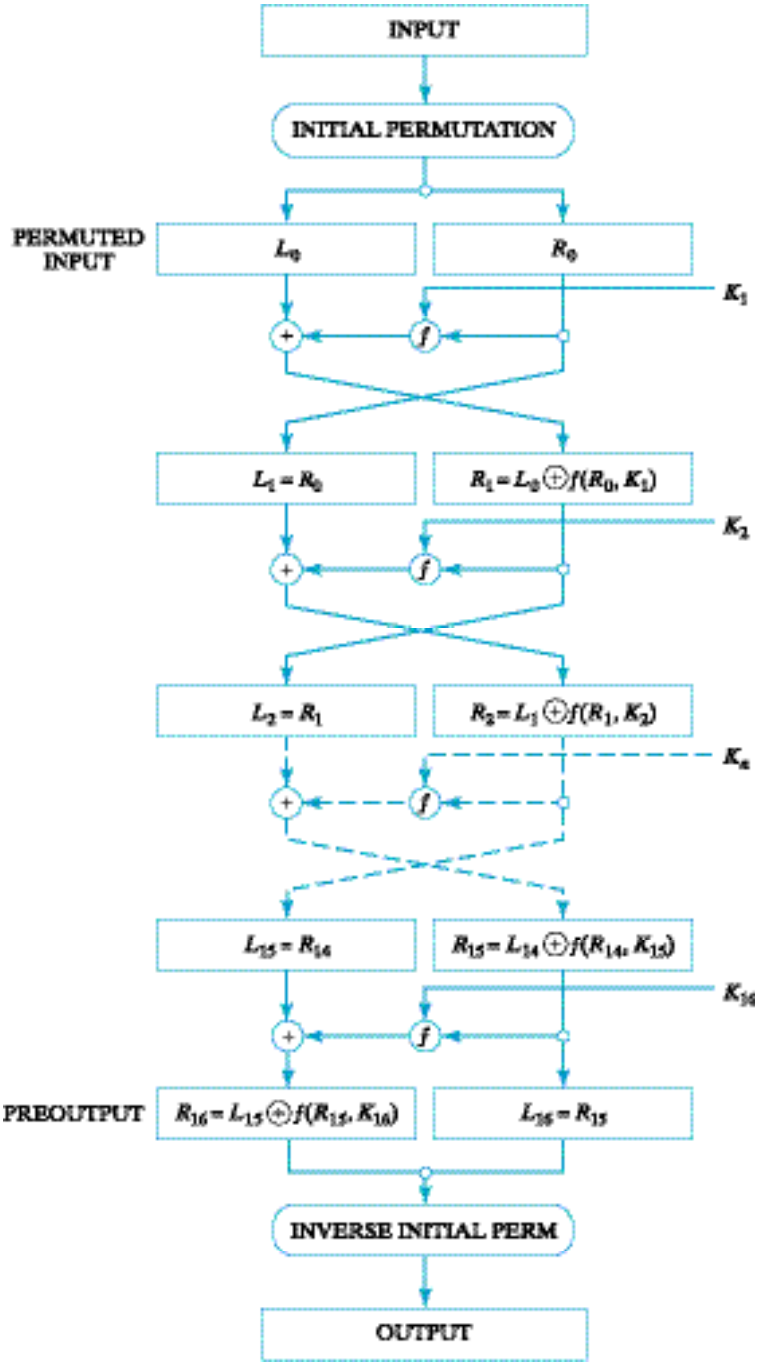
Original position:1 2 3
Permuted position:2 3 1

Data Encryption Standard

Most modern encryption schemes use both of these techniques, by combining several substitution and permutation operations. One of the best known is the **Data Encryption Standard (DES)** developed in the early 1970s by the federal government and the IBM corporation. The scheme is described in *Federal Information Processing Standards Publication 46* (FIPS Pub 46)[1] and is outlined in Figure 1, which is a diagram from this government publication.

1. Copies of this publication can be obtained from the National Institute of Standards and Technology of the U.S. Department of Commerce.

Figure 1. DES



In DES, the input is a bit string of length 64 representing a block of characters in the plaintext string (for example, the concatenation of the ASCII codes of eight characters), and the output is a 64-bit string that is the ciphertext. The encryption is carried out as a complicated series of permutations and substitutions. The substitution operations used are similar to those in earlier examples: Some are obtained by the addition of keywords and others use a substitution table.

The first operation applied to the 64-bit input string is an initial permutation (*IP*) given by the following table:

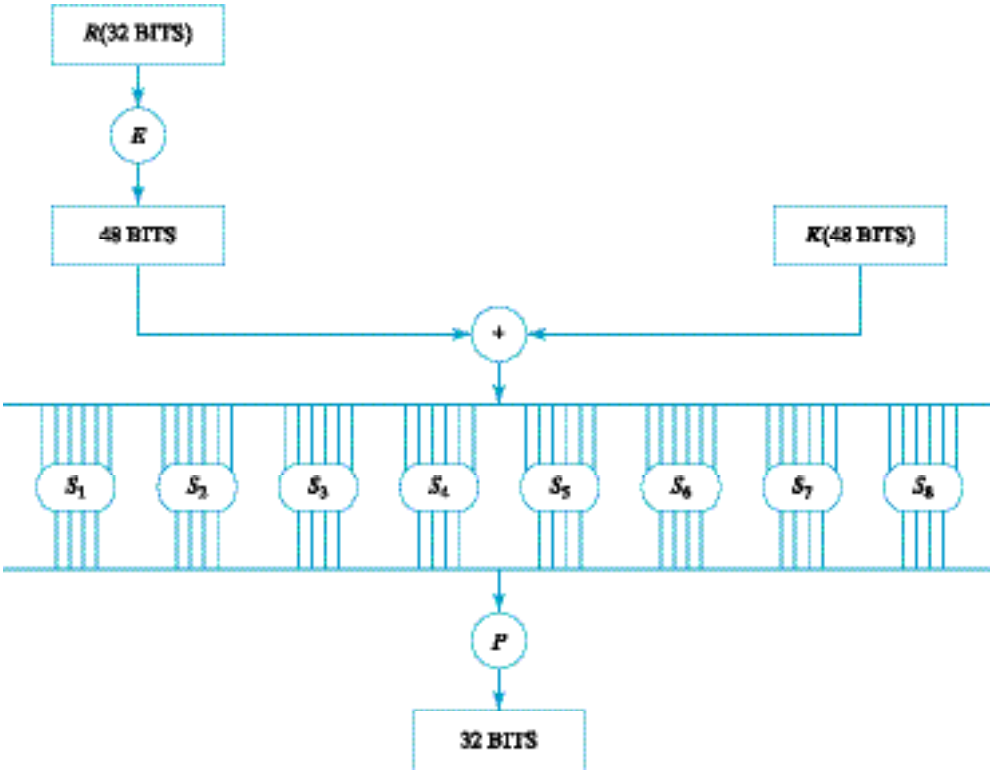
IP							
58	50	42	34	26	18	10	2
60	52	44	36	28	20	12	4
62	54	46	38	30	22	14	6
64	56	48	40	32	24	16	8
57	49	41	33	25	17	9	1
59	51	43	35	27	19	11	3
61	53	45	37	29	21	13	5
63	55	47	39	31	23	15	7

For example, the first bit in the permuted result is the fifty-eighth bit in the original string; the second bit is the fiftieth; and so on. This permuted string is then split into two 32-bit substrings: a left substring, denoted by L_0 in the diagram in Fig. 3.14, and a right substring, denoted by R_0. A **cipher function** denoted by f uses substitutions and a key K_1 to transform R_0 into a new 32-bit string denoted by $f(R_0,K_1)$. This string is then added to L_0 using bit-by-bit addition modulo 2 (that is, they are combined using the exclusive or operation) to produce the right substring R_1 at the next stage. The original R_0 becomes the left substring L_1.

This basic sequence of operations is performed 16 times with 16 different key strings $K_1, \ldots, K_{16}$, except that no "crossover" is performed at the last stage. These operations produce a 64-bit string $R_{16}L_{16}$ labeled "PREOUTPUT" in the diagram. The inverse of the initial permutation (IP^{-1}) is then applied to this preoutput string to yield the final ciphertext.

***IP*$^{-1}$**							
40	8	48	16	56	24	64	32
39	7	47	15	55	23	63	31
38	6	46	14	54	22	62	30
37	5	45	13	53	21	61	29
36	4	44	12	52	20	60	28
35	3	43	11	51	19	59	27
34	2	42	10	50	18	58	26
33	1	41	9	49	17	57	25

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



The details of the operation *f* are shown in Figure 2. The right substring denoted by *R* is first expanded into a 48-bit string using the following bit-selection table *E*:

E					
32	1	2	3	4	5
4	5	6	7	8	9
8	9	10	11	12	13
12	13	14	15	16	17
16	17	18	19	20	21
20	21	22	23	24	25
24	25	26	27	28	29
28	29	30	31	32	1

Thus the first 6-bit block consists of bits 32, 1, 2, 3, 4, and 5 of *R*; the second block consists of bits 4, 5, 6, 7, 8, and 9; and so on. A substitution operation is then applied to this 48-bit string by combining it with a 48-bit key string *K* using the exclusive or operation. Another substitution using a different table is then applied to each of the 6-bit blocks to produce 4-bit blocks so that the final result is again a 32-bit string. For example, the substitution table for S_1 is

S_1																
	Column Number															
Row No.	**0**	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**13**	**14**	**15**
0	14	4	13	1	2	15	11	8	3	10	6	12	5	9	0	7
1	0	15	7	4	14	2	13	1	10	6	12	11	9	5	3	8
2	4	1	14	8	13	6	2	11	15	12	9	7	3	10	5	0
3	15	12	8	2	4	9	1	7	5	11	3	14	10	0	6	13

To illustrate how it is used, suppose that the first 6-bit block is 101000. The binary numeral 10 consisting of the first and last bits determines a row in this table, namely, row 2, and the middle four bits 0100 determine a column, namely, column 4. The 4-bit binary representation 1101 of the entry 13 in the second row and the fourth column of this table is the replacement for this 6-bit block. Similar substitution tables S_2, . . ., S_8 are used to transform the other seven 6-bit blocks.

One final permutation P is applied to the resulting 32-bit string to yield $f(R, K)$:

P			
16	7	20	21
29	12	28	17
1	15	23	26
5	18	31	10
2	8	24	14
32	27	3	9
19	13	30	6
22	11	4	25

The 16 different keys used in DES are extracted in a carefully prescribed way from a single 64-bit key. Thus the user need supply only one key string to be used for encryption and decryption, rather than 16 different keys. The algorithm for decrypting ciphertext is the same as that for encryption, except that the 16 keys are applied in reverse order.

Because it was thought to be almost impossible to break, DES was adopted by the National Institute of Standards and Technology (formerly the National Bureau of Standards) as the standard encryption scheme for sensitive federal documents. It soon became one of the most widely-used methods of data encryption.

Questions about whether the 48-bit keys used in the substitutions were long enough and the substitution keys sophisticated enough to provide the necessary security were soon raised, however. And in 1998, DES was broken by the Electronic Frontier Foundation (EFF) using custom-designed chips and a personal computer running for 56 hours. Although it was reaffirmed in 1999 by the federal government as the encryption scheme of choice, it has since been replaced with a new standard for encryption known as the Advanced Encryption Standard (AES)

Public-Key Encryption

Each of the encryption schemes considered thus far requires that both the sender and the receiver know the key or keys used in encrypting the plaintext. This means that although the cryptogram may be transmitted through some public channel such as a telephone line that is not secure, the keys must be transmitted in some secure manner, for example, by a courier. This problem of maintaining secrecy of the key is compounded when it must be shared by several people.

Another popular type of encryption scheme eliminates this problem by using two keys, one for encryption and one for decryption. These schemes are called **public-key encryption schemes** because the encryption key is not kept secret. The keys used in these systems have the following properties:

1. For each encryption key there is exactly one corresponding decryption key, and it is distinct from the encryption key.
2. There are many such pairs of keys, and they are relatively easy to compute.
3. It is almost impossible to determine the decryption key if one knows only the encryption key.
4. The encryption key is made public by the receiver to all those who will transmit messages to him or her, but only the receiver knows the decryption key.

In 1978, Rivest, Shamir, and Adelman proposed one method of implementing a public key encryption scheme.[1] The public key is a pair (e, n) of integers, and one encrypts a message string M by first dividing M into blocks $M_1, M_2, \ldots, M_k$ and converting each block M_i of characters to an integer P_i in the range 0 through $n - 1$ (for example, by concatenating the ASCII codes of the characters). M is then encrypted by raising each block to the power e and reducing modulo n:

$$\text{Plaintext:} \quad = M_1 M_2 \quad M_k \rightarrow P_1 P_2$$

$$\text{Ciphertext:} \; = C_1 C_2 \quad C_k, C_i \; = P\% \, n$$

(Here % is the mod operator in C++.) The cipher text C is decrypted by raising each block C_i to the power d and reducing modulo n, where d is a secret decryption key. Clearly, to recover the plaintext, we need

$$P_i = C_i^d \% \, n = (P_i^e)^d \% \, n = P_i^{e.d} \% \, n$$

for each block P_i. Thus e and d must be chosen so that

$$x^{e \bullet d} \% \, n = x$$

for each nonnegative integer x.

The following algorithms summarize this Rivest-Shamir-Adelman (RSA) public key encryption system:

---

1. R. L. Rivest, A. Shamir, and L. Adelman, "A method for obtaining digital signatures and public-key cryptosystems," *communications of the ACM* 21, 2 (February 1978): 120–126

RSA Encryption Algorithm

/* This algorithm encrypts a plaintext using the RSA scheme with a public encryption code (*e*, *n*) to produce a ciphertext.

Receive: Plaintext *M*.

Return: Ciphertext *C*.

----------*/

1. Pad *M* with some randomly selected character if necessary so that *length*(*M*) is a multiple of *blockLength*.
2. Calculate *numberOfBlocks* = *length*(*M*) / *blockLength*.
3. Initialize index *j* to 1.
4. For *i* = 1 to *numberOfBlocks*:
   - **a.** Extract the substring M_i from *M* consisting of the *blockLength* characters beginning at position *j*.
   - **b.** Convert M_i to numeric form to give P_i.
   - **c.** Calculate $C_i = P_i^e \% n$
   - **d.** Increment *j* by *blockLength*.

RSA Decryption Algorithm

/* This algorithm decrypts a ciphertext using a secret decryption key *d* to produce a plaintext.

Receive: Ciphertext *C* consisting of numeric blocks C_i, $i = 1, \ldots,$ *numberOfBlocks*

Return: Plaintext *M*.

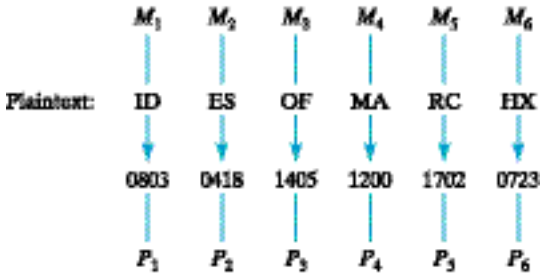
----------*/

1. Initialize *M* to the empty string.
2. For *i* = 1 to *numberOfBlocks*:
   - **a.** Calculate $P_i = C_i^d \% n$
   - **b.** Convert P_i to a string of characters M_i.
   - **c.** Concatenate M_i onto *M*.

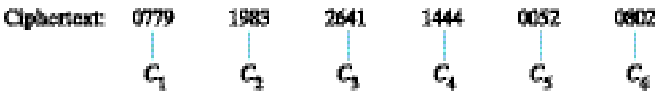
To illustrate, suppose that (17, 2773) is the public encryption code and that characters are converted to numeric values using the following table:

Character:	A	B	C	D	E	F	G	H	I	J	K	L	M
code:	00	01	02	03	04	05	06	07	08	09	10	11	12
Character:	N	O	P	Q	R	S	T	U	V	W	X	Y	Z
code:	13	14	15	16	17	18	19	20	21	22	23	24	25

To encrypt a string such as M = "IDESOFMARCH" using the RSA algorithm, we divide M into 2-character blocks $M_1, M_2, \ldots, M_6$ (after appending the randomly selected character X) and represent each block M_i as an integer P_i in the range 0 through 2773 – 1 = 2772 by concatenating the numeric codes of the characters that comprise the block:



Each of these blocks P_i is then encrypted by calculating $C_i = P_i^{13} \% 2773$:



For this encryption key, the corresponding decrypting key is d = 157. Thus, we decrypt the ciphertext by calculating $C_i^{157} \% 2773$ for each block C_i. For the preceding ciphertext this gives

Decrypted ciphertext: 0803 0418 1405 1200 1702 0723

which is the numeric form of the original message.

Two points in the preceding discussion of the RSA encryption scheme require further explanation: (1) How are n, e, and d chosen? (2) How can the exponentiation be performed efficiently?

The number n is the product of two large "random" primes p and q,

$$n = p \cdot q$$

In the preceding example, we used the small primes 47 and 59 to simplify the computations, but Rivest, Shamir, and Adelman suggest that p and q have at least one hundred digits. The decrypting key d is then selected to be some large random integer that is relatively prime to both $p - 1$ and $q - 1$, that is, one that has no factors in common with either number. In our example, d = 157 has this property. The number e is then selected to have the property that

$$e \cdot d \% [(p - 1) \cdot (q - 1)] \text{is equal to } 1$$

A result from number theory then guarantees that e and d will have the required property described earlier, namely, that

$$P_i^{e \bullet d} \% n = P_i$$

for each block P_i.

We can efficiently carry out the exponentiations required in encryption and decryption by repeatedly squaring and multiplying, as follows:

Exponentiation Algorithm

/* Algorithm to calculate $y = x^k \bmod n$.

Input: Integers x, k, and n.
Return: y.
----------*/

1. Find the base-2 representation $b_t \cdots b_1 b_0$ of the exponent k.
2. Initialize y to 1.
3. For $i = t$ down to 0:
   - **a.** Set $y = y^2 \% n$.
   - **b.** If $b_i = 1$ then
     Set $y = (y * x) \% n$.

Recall that the encrypting key (e, n) is a public key, so that no attempt is made to keep it secret. The decrypting key d is a private key, however, and so must be kept secret. To break this code, one would need to be able to determine the value of d from the values of n and e. Because of the manner in which d and e are selected, this is possible if n can be factored into a product of primes. The security of the RSA encryption scheme is based on the difficulty of determining the prime factors of a large integer. For large prime factors, this is a prohibitively time-consuming task. A study of a few years ago gave the following table displaying some estimated times, assuming that each operation required one microsecond.

Number of Digits in Number Being Factored	Time
50	4 hours
75	104 days
100	74 years
200	4 billion years
300	$5 \infty 10^{15}$ years
500	$4 \infty 10^{25}$ years

EXERCISES

1. A pure permutation encryption scheme is very insecure. Explain why by describing how an encryption scheme that merely permutes the bits in an n-bit string can easily be cracked by studying how certain basic bit strings are encrypted. Illustrate for $n = 4$.
2. Consider a simplified DES scheme that encrypts messages using the DES approach pictured in Fig. 3.14 but with only two keys, K_1 and K_2, instead of 16 keys $K_1, \ldots, K_{16}$, and that in the calculation of $f(R, K)$ pictured in Fig. 3.15 uses the same substitution table S_1 for each of the 6-bit blocks instead of eight different tables $S_1, \ldots, S_8$. Encrypt the string "AARDVARK" using this simplified DES scheme with keys K_1 = "ABCDEF" and K_2 = "SECRET" and assuming that strings are converted into bit strings by replacing each character by its binary ASCII codes.
3. Using the character codes 00, 01, . . ., 25 given in the text
   a) Find the RSA ciphertext produced by the key $(e, n) = (5, 2881)$ for the plaintext "PUBLIC."
   b) Verify that $d = 1109$ is a decrypting key for the RSA scheme in (**a**).
4. If the RSA ciphertext produced by key $(e,n) = (13, 2537)$ is 0095 and the character codes 00, 01, . . ., 25 given in the text are used, find the plaintext.
5. A public key encryption scheme can be used to provide positive identification of the sender of a message by incorporating a *digital signature* into it. To illustrate, suppose that Al wishes to send a message M to Bob. Al first "signs" M by encrypting it using his secret decrypting key, which we might indicate by
$$S = D_{\underline{\text{Al}}}(M)$$
6. He then encrypts S using Bob's public encryption key and sends the result to Bob:
$$M' = E_{\text{Bob}}(S)$$
7. Bob first decrypts the ciphertext M' with his secret decrypting key to obtain the signature S
$$D_{\text{Bob}}(M') = D_{\text{Bob}}(E_{\text{Bob}}(S)) = S$$
8. and then extracts the message M by using Al's public encryption key:
$$E_{\text{Al}}(S) = E_{\text{Al}}(D_{\text{Al}}(M)) = M$$
9. Bob's pair (M, S) is similar to a paper document that Al signed, since only Al could have created S. For the message M = "HI" and using the character codes 00, 01, . . ., 25 given in the text, find M' if Al and Bob have published RSA encryption keys (3, 1081) and (1243, 1829), respectively.
10. Write a function to implement the algorithm given in the text for calculating $y = x^k \% n$ by repeated squaring and multiplication.