DES & RSA

DES

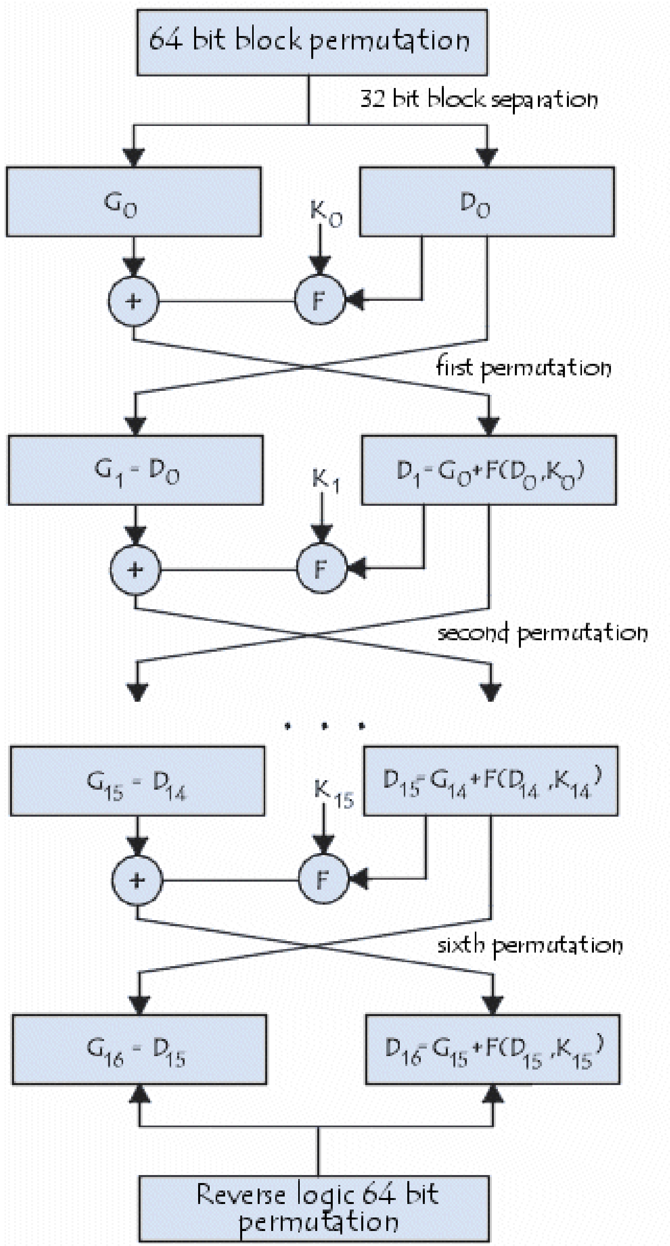
- In 1974, IBM proposed "**Lucifer**", which, thanks to the NSA (National Security Agency), was modified on 23 November 1976 to become the **DES** (Data Encryption Standard).
- The DES was approved by the NBS (National Bureau of Standards, now called NIST - National Institute of Standards and Technology) in 1978.
- The DES was standardized by the ANSI (American National Standard Institute) under the name of ANSI X3.92, better known as DEA (Data Encryption Algorithm).

DES Algorithm

The main parts of the algorithm are as follows:

- Fractioning of the text into 64-bit (8 octet) blocks;
- Initial permutation of blocks;
- Breakdown of the blocks into two parts: left and right, named L and R;
- Permutation and substitution steps repeated 16 times (called rounds);
- Re-joining of the left and right parts then inverse initial permutation.

DES Algorithm



DES Algorithm

- Initial permutation: each bit of a block is subject to initial permutation, which can be represented by the following initial permutation (IP) 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

DES Algorithm

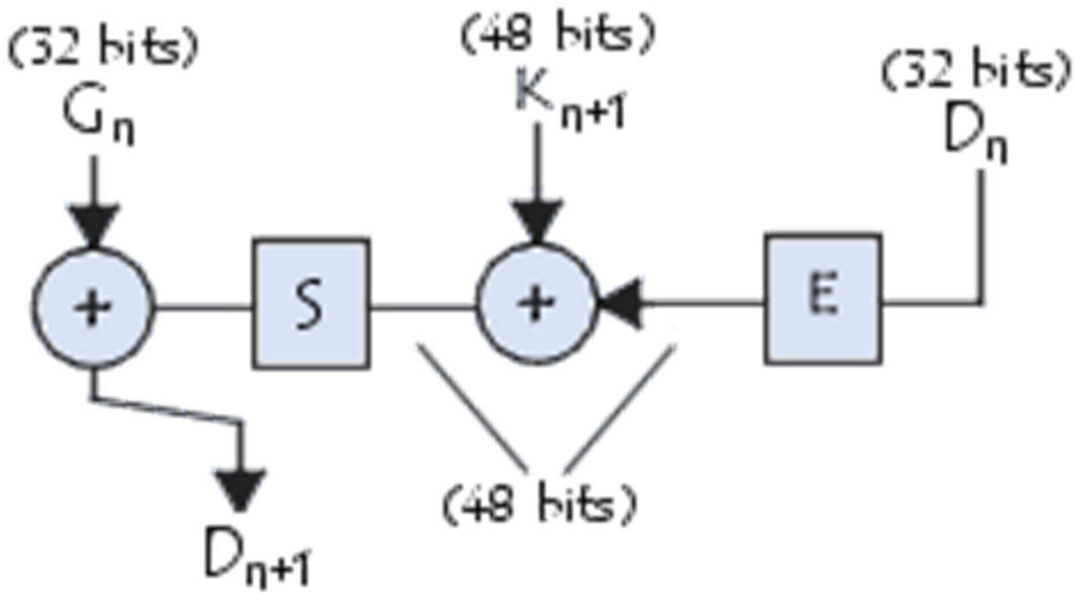
- Division into 32-bit blocks

L_0	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

R_0	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

DES Algorithm

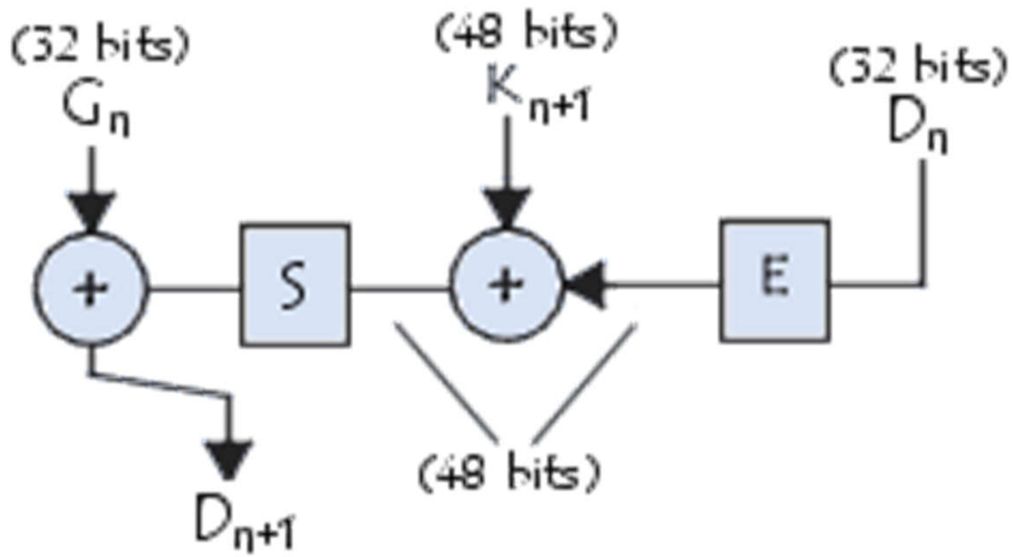
- Rounds: The Ln and Rn blocks are subject to a set of repeated transformations called rounds, shown in this diagram, and the details of which are given below:



DES Algorithm

- Expansion function: The 32 bits of the R_0 block are expanded to 48 bits thanks to a table called an expansion table (denoted 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
E	16	17	18	19	20	21
	20	21	22	23	24	25
	24	25	26	27	28	29
	28	29	30	31	32	1



DES Algorithm

- Substitution function:
  - R_0 is then divided into 8 6-bit blocks, denoted R_{0i}. Each of these blocks is processed by selection functions (sometimes called substitution boxes or compression functions), generally denoted S_i.
  - The first and last bits of each R_{0i} determine (in binary value) the line of the selection function; the other bits (respectively 2, 3, 4 and 5) determine the column

S_1	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

DES Algorithm

- Substitution function:
  - Let R_{01} equal 101110. The first and last bits give 10, that is, 2 in binary value. The bits 2,3,4 and 5 give 0111, or 7 in binary value. The result of the selection function is therefore the value located on line no. 2, in column no. 7. It is the value 11, or 111 binary.
  - Each of the 8 6-bit blocks is passed through the corresponding selection function, which gives an output of 8 values with 4 bits each

		0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
S_1	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

DES Algorithm

- Substitution function:

		0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
	0	15	1	8	14	6	11	3	4	9	7	2	13	12	0	5	10
S_2	1	3	13	4	7	15	2	8	14	12	0	1	10	6	9	11	5
	2	0	14	7	11	10	4	13	1	5	8	12	6	9	3	2	15
	3	13	8	10	1	3	15	4	2	11	6	7	12	0	5	14	9

		0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
	0	10	0	9	14	6	3	15	5	1	13	12	7	11	4	2	8
S_3	1	13	7	0	9	3	4	6	10	2	8	5	14	12	11	15	1
	2	13	6	4	9	8	15	3	0	11	1	2	12	5	10	14	7
	3	1	10	13	0	6	9	8	7	4	15	14	3	11	5	2	12

		0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
	0	7	13	14	3	0	6	9	10	1	2	8	5	11	12	4	15
S_4	1	13	8	11	5	6	15	0	3	4	7	2	12	1	10	14	9
	2	10	6	9	0	12	11	7	13	15	1	3	14	5	2	8	4
	3	3	15	0	6	10	1	13	8	9	4	5	11	12	7	2	14

		0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
	0	2	12	4	1	7	10	11	6	8	5	3	15	13	0	14	9
S_5	1	14	11	2	12	4	7	13	1	5	0	15	10	3	9	8	6
	2	4	2	1	11	10	13	7	8	15	9	12	5	6	3	0	14
	3	11	8	12	7	1	14	2	13	6	15	0	9	10	4	5	3

		0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
	0	12	1	10	15	9	2	6	8	0	13	3	4	14	7	5	11
S_6	1	10	15	4	2	7	12	9	5	6	1	13	14	0	11	3	8
	2	9	14	15	5	2	8	12	3	7	0	4	10	1	13	11	6
	3	4	3	2	12	9	5	15	10	11	14	1	7	6	0	8	13

		0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
	0	4	11	2	14	15	0	8	13	3	12	9	7	5	10	6	1
S_7	1	13	0	11	7	4	9	1	10	14	3	5	12	2	15	8	6
	2	1	4	11	13	12	3	7	14	10	15	6	8	0	5	9	2
	3	6	11	13	8	1	4	10	7	9	5	0	15	14	2	3	12

		0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
	0	13	2	8	4	6	15	11	1	10	9	3	14	5	0	12	7
S_8	1	1	15	13	8	10	3	7	4	12	5	6	11	0	14	9	2
	1	7	11	4	1	9	12	14	2	0	6	10	13	15	3	5	8
	1	2	1	14	7	4	10	8	13	15	12	9	0	3	5	6	11

DES Algorithm

- Permutation: The obtained 32-bit block is then subject to a permutation P here is the table:

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

- Exclusive OR: All of these results output from P are subject to an Exclusive OR with the starting L_0 (as shown on the first diagram) to give R_1, whereas the initial R_0 gives L_1.
- Iteration: All of the previous steps (rounds) are repeated 16 times.

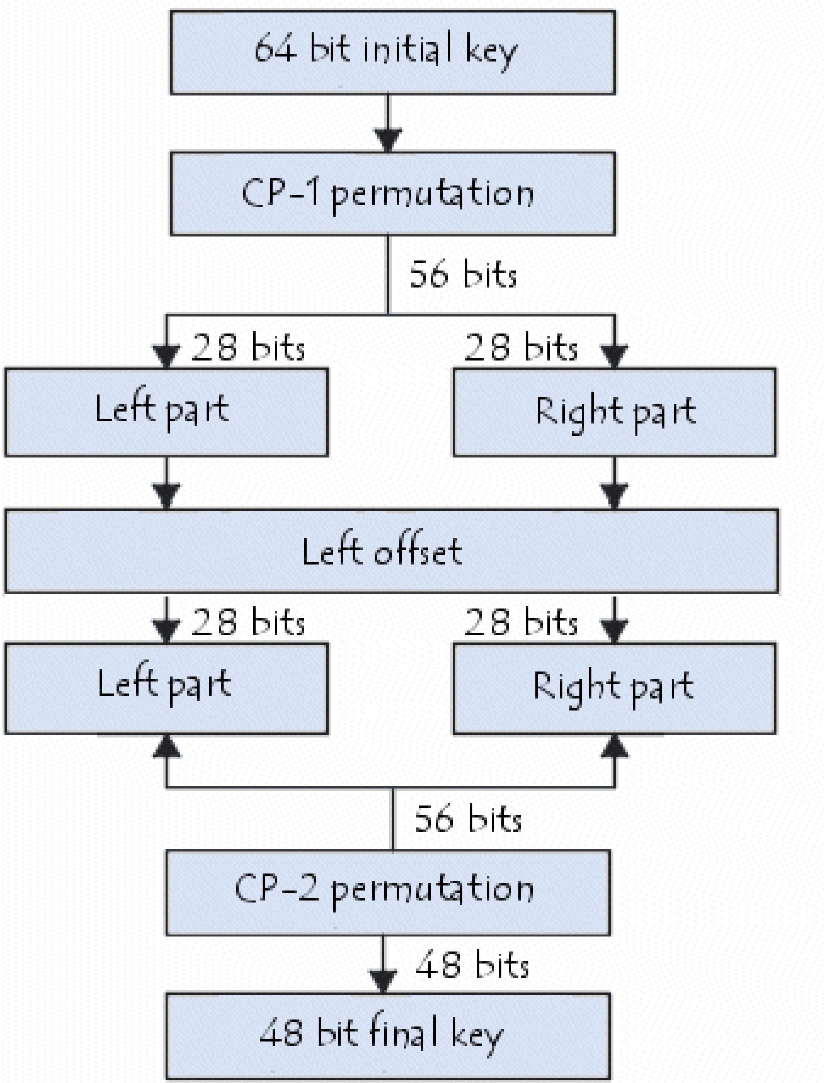
DES Algorithm

- Inverse initial permutation: At the end of the iterations, the two blocks L_{16} and R_{16} are re-joined, then subject to inverse initial permutation, the output result is a 64-bit 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

DES Algorithm

- Generation of keys:



DES Algorithm

- Generation of keys:

PC-1

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

L_i

57	49	41	33	25	17	9
1	58	50	42	34	26	18
10	2	59	51	43	35	27
19	11	3	60	52	44	36

R_i

63	55	47	39	31	23	15
7	62	54	46	38	30	22
14	6	61	53	45	37	29
21	13	5	28	20	12	4

DES Algorithm

- Generation of keys:
  - The result of this first permutation is denoted L_0 and R_0.
  - These two blocks are then rotated to the left
  - The 2 28-bit blocks are then grouped into one 56-bit block. This passes through a permutation, denoted PC-2, giving a 48-bit block as output, representing the key K_i.

PC-2	14	17	11	24	1	5	3	28	15	6	21	10
	23	19	12	4	26	8	16	7	27	20	13	2
	41	52	31	37	47	55	30	40	51	45	33	48
	44	49	39	56	34	53	46	42	50	36	29	32

DES Algorithm

- Generation of keys:
  - Repeating the algorithm makes it possible to give the 16 keys K_1 to K_{16} used in the DES algorithm.

Iteration Number	Number of Left Shifts
1	1
2	1
3	2
4	2
5	2
6	2
7	2
8	2
9	1
10	2
11	2
12	2
13	2
14	2
15	2
16	1

DES Algorithm - Example

- http://page.math.tu-berlin.de/~kant/teaching/hess/krypto-ws2006/des.htm

RSA

- RSA stands for Ron Rivest, Adi Shamir and Leonard Adleman, who first publicly described the algorithm in 1977
- The RSA algorithm involves three steps: key generation, encryption and decryption.

RSA

Key generation:

- Choose two distinct prime numbers p and q (random)
- Compute $n = pq$.
- Compute $\phi(n) = (p - 1)(q - 1)$
- Choose an integer e such that $1 < e < \phi(n)$ and $\gcd(e, \phi(n)) = 1$; i.e., e and $\phi(n)$ are coprime. e is released as the public key exponent.
- Determine d as $d \equiv e^{-1} \pmod{\phi(n)}$; d is kept as the private key exponent. ($e.d = 1 \bmod \phi(n)$)

Public key: n & e

Private key: n & d

RSA

Encryption:

- Alice transmits her public key (n, e) to Bob and keeps the private key d secret.
- Bob then wishes to send message M to Alice, he first turns M into an integer m, such that $0 \le m < n$. He then computes the ciphertext c corresponding to:

$$c \equiv m^e \pmod{n}$$

RSA

Decryption:

- Alice can recover m from c by using her private key exponent d via computing:

$$m \equiv c^d \pmod{n}$$

RSA - Example

Lets choose two primes: $p=11$ and $q=13$.

Hence the modulus is $n=p\times q=143$. $\phi(n)=(p-1)\cdot(q-1)=120$.

Choose any number $1<e<120$ that is coprime to 120. Let $e = 7$

Compute $d = 103$.

$$e \times d \bmod \phi(n) = 1$$

$$7 \times 103 \bmod \phi(120) = 1$$

Public key is $(n=143, e=7)$

Private key is $(n=143, d=103)$

RSA - Example

Public key is (n=143, e=7)

Private key is (n=143, d=103)

For example, plaintext m = 9

Encryption: $m^e \bmod n = 9^7 \bmod 143 = 48 = c$

Decryption: $c^d \bmod n = 48^{103} \bmod 143 = 9 = m$

RSA - Example

Real example:

p

121310724392112718973236715316124404284724276337014109256345493123019643730420856193241973653
224168665410170573613652141717117137979742993348710628298035 41

q

120275242554787488859562207937345121287333878036820754336538999839551798509887978998691469008
091316111533468170508320960221601463663463918124709871054152 33

With these two large numbers, we can calculate n and ϕ(n)

n

145906768007583323230186939349070635292401872375357164399581871019873438799005358938369571402
670149802121818086292467422828157022922076746906543401224889672472407926969987100581290103199
317858753663710862357656510507883714297115637342788911463535102712032765166518411726859837988
672111837205085526346618740053

ϕ(n)

145906768007583323230186939349070635292401872375357164399581871019873438799005358938369571402
670149802121818086292467422828157022922076746906543401224889648313811232279966317301397777852
365301547848273478871297222058587457152891606459269718119268971163555070802643999529549644116
811947516513938184296683521280

e - the public key

65537 has a gcd of 1 with ϕ(n), so lets use it as the public key. To calculate the private key, use extended euclidean algorithm to find the multiplicative inverse with respect to ϕ(n).

d - the private key

894894250092744443682285459217730939196695860658842574454978544564876748396298183909349419732
628796167979706089172836798754993315741611138540888132754881105882471930775825272784379065040
156806234235500672400424666656542323835029222154936232894721388664458187891279461234078077257
026266440910365023725451 39713

RSA - Example

Plaintext: “attack at dawn” to decimal
1976620216402300889624482718775150

Encryption: $1976620216402300889624482718775150^{e} \bmod n$

35052111338673026690212423937053328511880760811579981620642802346685810623109850235943049080973386241113784040794704193978215378499765413083646438784740952306932534945195080183861574225226218879827232453912820596886440377536082465681750074417459151485407445862511023472235560823053497791518928820272257787786

Decryption:

$35052111338673026690212423937053328511880760811579981620642802346685810623109850235943049080973386241113784040794704193978215378499765413083646438784740952306932534945195080183861574225226218879827232453912820596886440377536082465681750074417459151485407445862511023472235560823053497791518928820272257787786^{d} \bmod n$

Reference

- http://en.wikipedia.org/wiki/Data_Encryption_Standard
- http://en.wikipedia.org/wiki/RSA_(cryptosystem)
- http://doctrina.org/How-RSA-Works-With-Examples.html
- http://www.heliwave.com/RSADemo.html
- http://asecuritysite.com/encryption/rsa
- http://tizhoosh.uwaterloo.ca/Teaching/RSA.htm