RSA algorithm

RSA Public-Key Encryption Algorithm

- One of the first, and probably best known public-key scheme;
- It was developed in 1977 by R.Rivest, A.Shamir and L. Adleman;
- RSA is a block cipher in which the plaintext and ciphertext are **integers** between **0** and **n-1**, where
- **n** is some number;
- Every integer can be represented, of course, as a sequence of bits;

Encryption and decryption in RSA

- **Encryption**
- $$C = M^e \bmod n$$
- **Decryption**

$$M = C^d \bmod n = (M^e)^d \bmod n = M^{ed} \bmod n$$

Here M is a block of a plaintext, C is a block of a ciphertext and e and d are some numbers. Sender and receiver know *n* and *e*. Only the receiver knows the value of *d*.

Private and Public keys in RSA

- Public key KU = {e,n};
- Private key KR = {d,n};
- **Requirements:**
- It is possible to find values *e,d,n* such that

$$M^{ed} = M \bmod n \text{ for all } M < k$$

- It is easy to calculate M^e and C^d modulo n
- It is difficult to determine *d* given *e* and *n*

Key generation

- Select two prime numbers *p* and *q;*
- Calculate $n = p \times q$;
- Calculate $\phi(n)$ = (p-1)(q-1);
- Select integer *e* less than $\phi(n)$ and relatively prime with $\phi(n)$;
- Calculate *d* such that $de \bmod \phi(n) = 1$
- Public key *KU* = *{e,n}*;
- Private key *KR* = *{d,n}*;

Fermat – Euler Theorem

- Correctness of RSA can be proved by using Fermat-Euler theorem:

$$x^{p-1} = 1 \bmod p$$

Where *p* is a prime number *and* $x \neq 0 \bmod p$

Chinese Remainder Theorem

For relatively prime p and q and any x and y

$$x = y \bmod p$$
$$x = y \bmod q$$

Implies

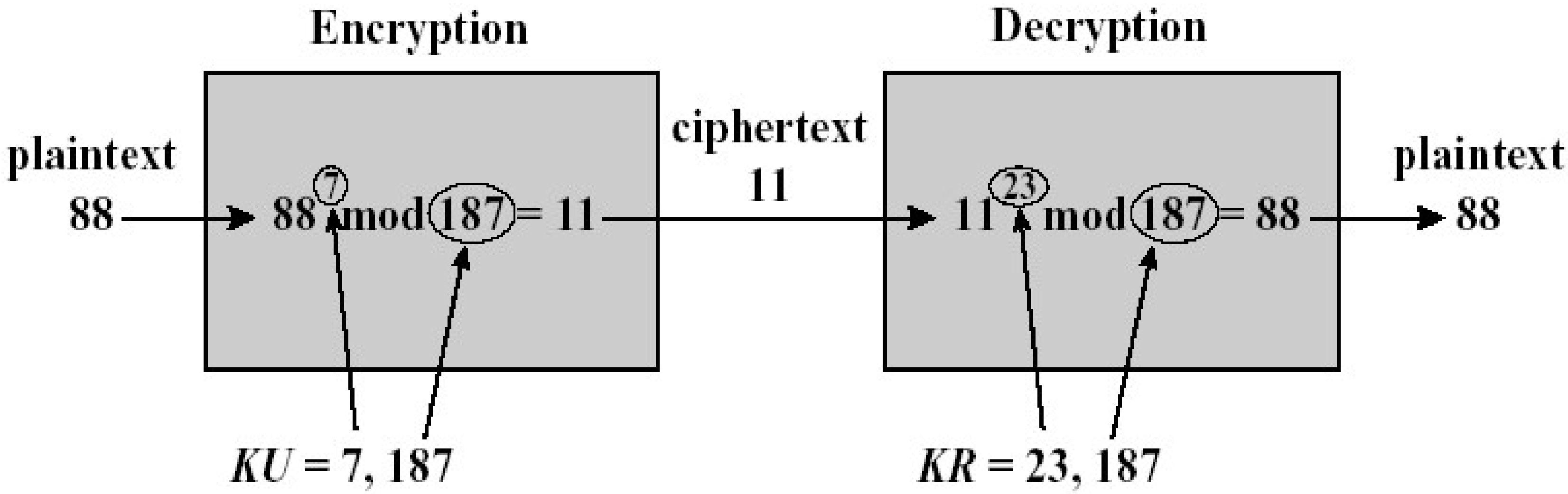
$$x = y \bmod pq$$

Example

- Select two prime numbers, $p = 17$, $q = 11$;
- Calculate $n = pq = 187$;
- Calculate $\phi(n) = 16 \times 10 = 160$;
- Select e less than 160 and relatively prime with 160, for example 7 ;
- Determine d such that $de \bmod 160 = 1$ and $d < 160$. The correct value is $d = 23$, indeed $23 \times 7 = 161 = 1 \bmod 160$.
- Thus KU = {7,187} and KR = {23,187} in that case.

Encryption and decryption

- Let a plaintext be M = 88; then encryption with a key {7,187} and decryption with a key {23,187} go as follows



How to break RSA

- **Brute-force approach**: try all possible private keys of the size *n*. Too many of them even for moderate size of *n;*

- **More specific approach**: given a number *n,* try to find its two prime factors *p* and *q;* Knowing these would allow us to find a private key easily*.*

Security of RSA

- Relies upon complexity of factoring problem:
- Nobody knows how to factor the big numbers in the reasonable time (say,in the time polynomial in the size of (binary representation of) the number;
- On the other hand nobody has shown that the fast factoring is impossible;

RSA challenge

- RSA Laboratories to promote investigations in security of RSA put a challenge to factor big numbers. Least number, not yet factored in that challenge is

- RSA-232 =
  1009881397871923546909564894309468582818233821955573955141120516
  2058310213385285453743661097571543636649133800849170651699217015
  2473329438927028023438096090980497644054071120196541074755382494
  8672771374075011577182305398340606162079

- 768 bits, or 232 decimal digits

RSA challenge, very recent news

RSA-230 =

1796949159794106673291612844957324615636756180801260007088891883553172 6
4603414909334933722478686507552308558641999292218144366847228740520652 5
7937495694348389263171152522525654410980819170611742509702440718010364 8
316382 88518852689 =

45284503580104920266124397391201667589112460474937000400739567592615903
97 2500336993576945071935230003430886016885 89

X

39681326231509575885323944390498873417695339666219578294269660840930495
16 953598120833228447171744337427374763106901

230 decimal digits (762 bits)

(S. Gross et al, Noblis Inc., August, 2018)

How to break RSA (cont.)

- Common factors attack (2012):

due to insufficiently good random number generators used in key generation, some amount of keys used in the wild have common divisors - you can then factorized them using ECD (Euclid Common Divisor algorithm)

- Shor's factorization algorithm for quantum computers (near future?)