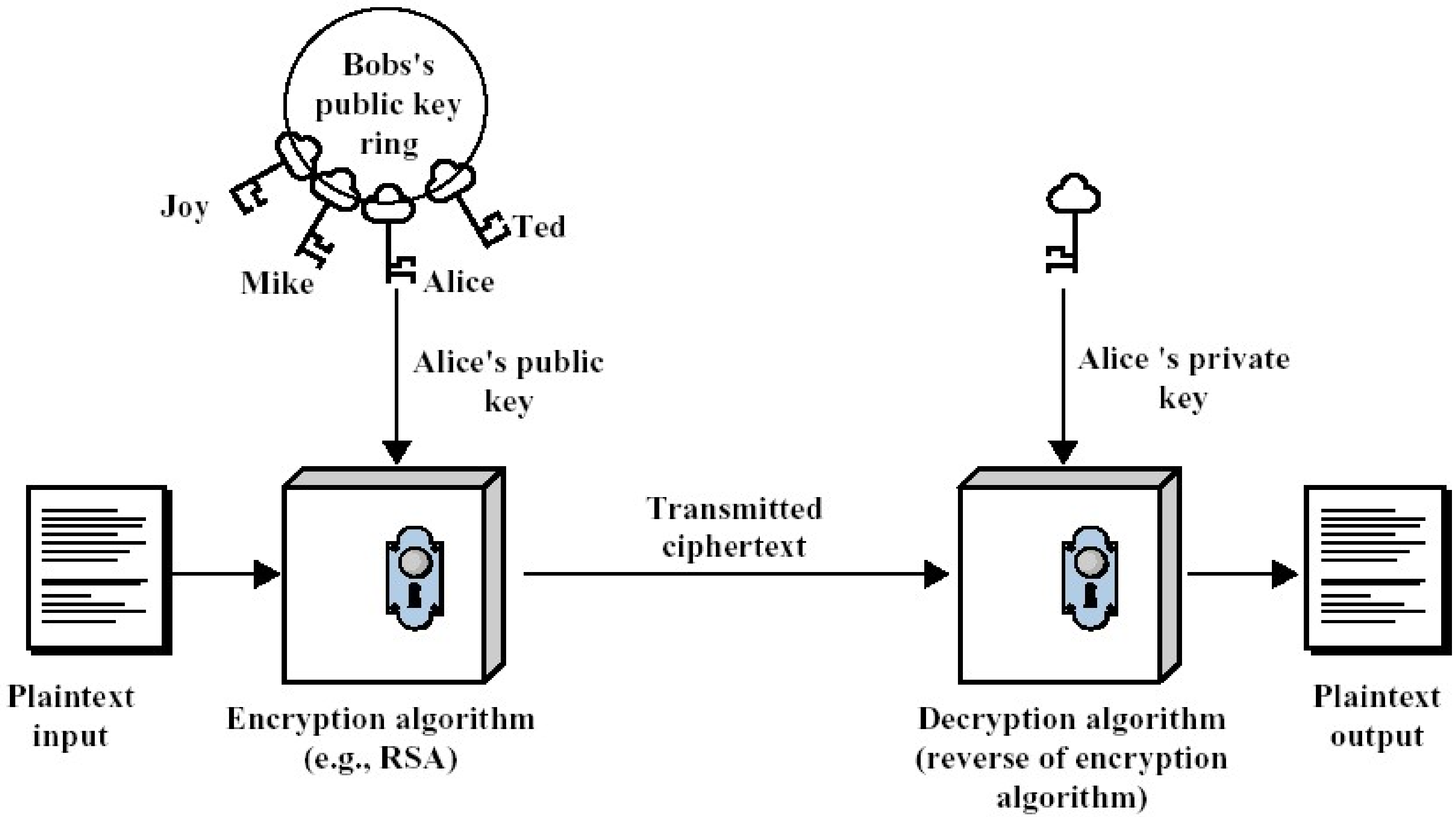
Public-Key Encryption

Public-key, or asymmetric encryption

- **Public-key encryption** techniques. It is particular and most important kind of

- **Asymmetric encryption** (or asymmetric key encryption):
  - **One key** is used for encryption (usually publicly known, *public key*);
  - **Another key** is used for decryption (usually *private*, or *secret* key)

Public-key encryption



(a) Encryption

Components of public-key encryption

- Plaintext
- Encryption algorithm
- Public and private key
- Ciphertext
- Decryption algorithm

Essential steps in communications using public-key encryption

- Each user generates a **pair** of keys;
- Each users makes one of the key publicly accessible (public key). The other key of the pair is kept private;
- If B wishes to send a private message to A, B **encrypts** the message using **A's public key;**
- When A receives the message, A **decrypts** it using **A's private key**. No other recipient can decrypt the message – nobody else knows A's private key.

Public-key encryption

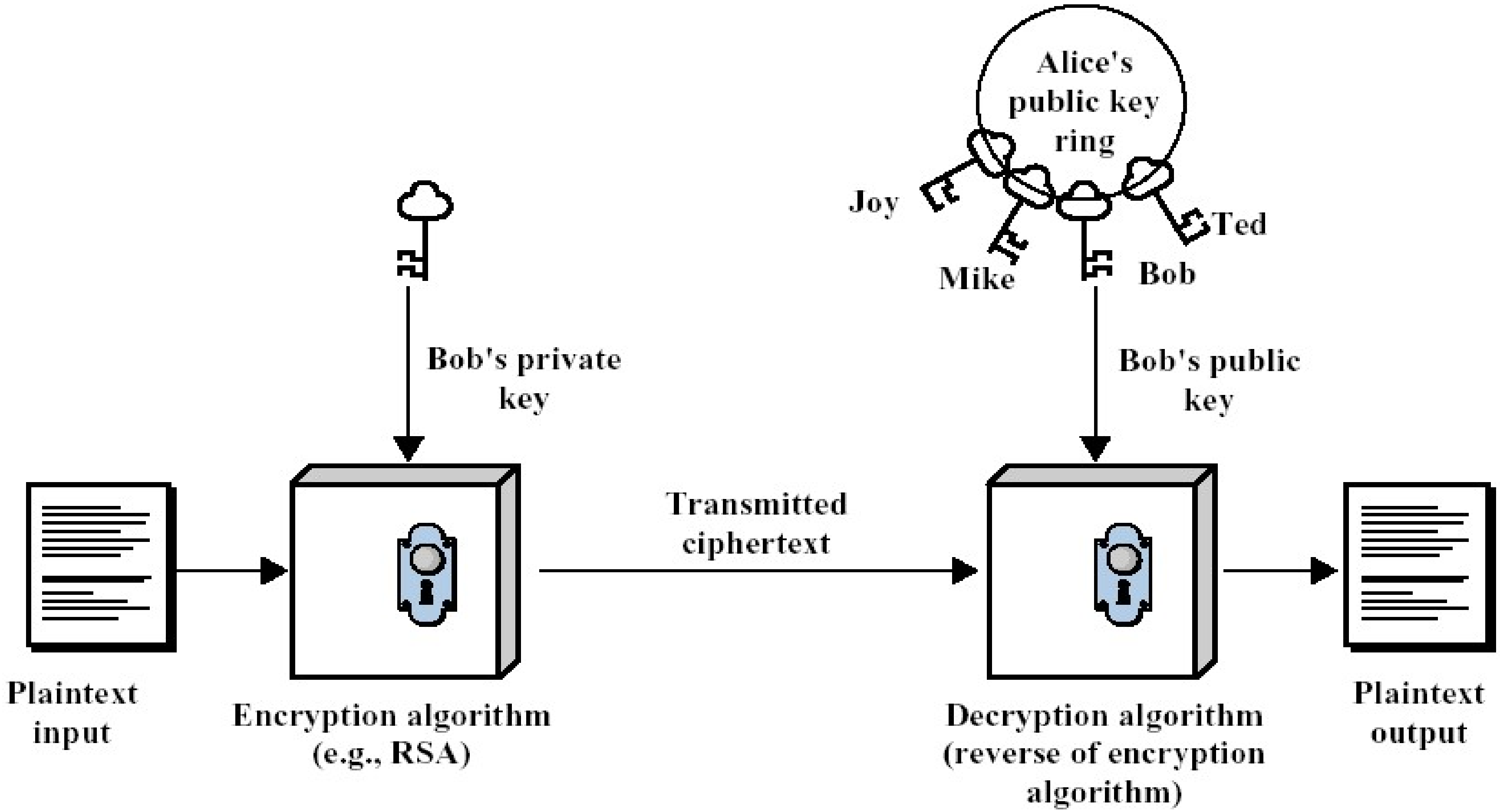
- **Advantages**
- All keys (public and private) are generated locally;
- No need in distribution of the keys;
- Moreover, each user can change his own pair of public/private key at any time;

- **Disadvantages**
- It is more computationally expensive.

Applications of Public-Key Cryptosystems

- **Encryption/decryption:** the sender encrypts a message with the recipient's public key.
- **Digital signature (authentication):** the sender "signs" the message with its private key; a receiver can verify the identity of the sender using sender's public key.
- **Key exchange:** both sender and receiver cooperate to exchange a (session) key.

Authentication using public-key systems



(b) Authentication

Requirements for Public-Key Cryptography

- **Diffie and Hellman conditions**
- **“Easy part”**
- It is computationally easy for a party B to generate a pair (public key , private key).
- It is computationally easy for a sender A, knowing the public key of B and the message M to generate a ciphertext:
- It is computationally easy for the receiver B to decrypt the resulting ciphertext using his private key

Requirements for Public-Key Cryptography

- **“Difficult part”**
- It is computationally infeasible for anyone, knowing the public key, to determine the private key,

- **Additional useful requirement** (not always necessary)
- Either of the two related keys can be used for encryption, with the other used for decryption.

-

Public-key cryptography and number theory

- Many public-key cryptosystems use non-trivial number theory;
- Security of most known RSA public-key cryptosystem is based on the hardness of factoring big numbers;

- We will overview basic notions of divisors, prime numbers, modular arithmetic

Divisors and prime numbers

- **Divisors**
- Let **a** and **b** are integers and **b** is not equal to **0;**
- then we say **b** is a divisor of **a** if there is an integer **m** such that **a** = **mb**;

- **Prime numbers**
- An integer **p** is a *prime number* if its only divisors are **1, -1, p, -p**

gsd and relatively prime numbers

- **gcd(a,b)** is a greatest common divisor of **a** and **b**
- Examples: gcd(12, 15) = 3; gcd(49,14) = 7.

- **a** and **b** are **relatively prime** if gcd(a,b) = 1.
- Example: gcd (9,14) = 1.

Modular arithmetic

- If ***a*** is an integer and ***n*** is a positive integer, we define ***a mod n*** to be the remainder when *a* is divided by *n:*
- ***a = qn+r,***
- Here ***q*** is a quotient and ***r = a mod n***
- If ***(a mod n) = (b mod n)*** then ***a*** and ***b*** are ***congruent modulo n***;
- It is easy to see, that ***(a mod n) = (b mod n)*** iff ***n*** is a divisor of ***a-b***.

Modular arithmetic. Properties

- ***[(a mod n) + (b mod n)] mod n = (a+b) mod n***
- ***[(a mod n) – (b mod n)] mod n = (a-b) mod n***
- ***[(a mod n) x (b mod n)] mod n = (a x b) mod n***
- Example: 3 mod 5 x 4 mod 5 = 12 mod 5 = 2 mod 5

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
-
- It is easy to calculate

Requirements

- It is possible to find values *e,d,n* such that

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

- (key generation) , where k is some number , k < n

- 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* x *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

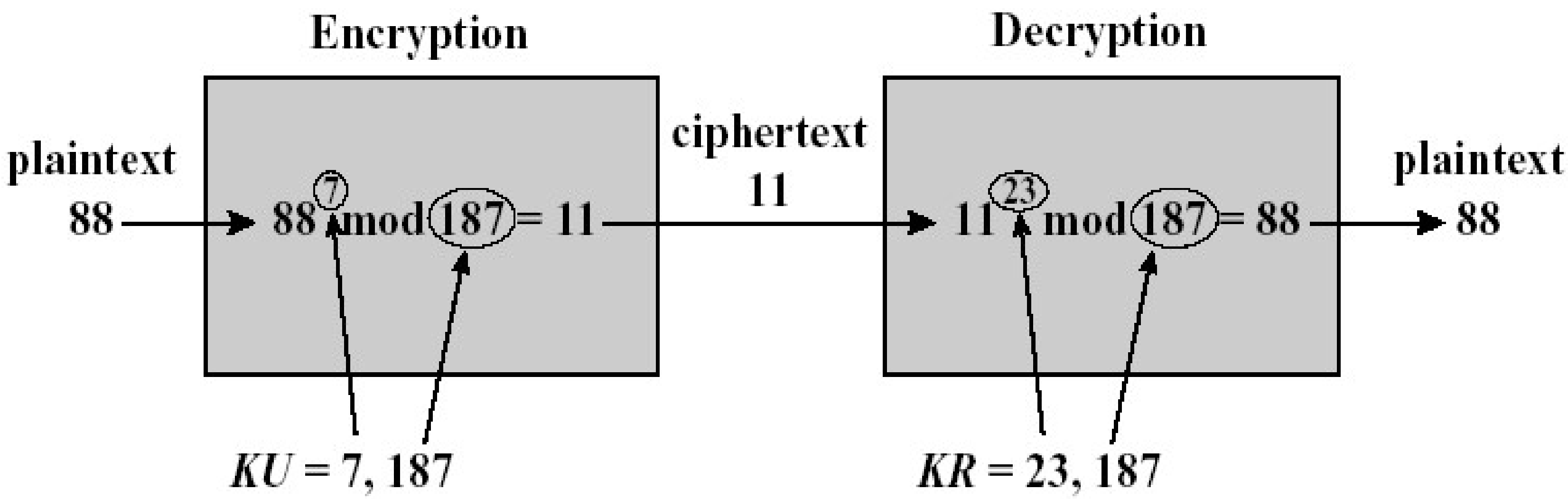
$$x = y \bmod pq$$

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;
- Let e = 7;
- Determine *d* such that *de* mod 160 = 1 and d < 160. The correct value is *d* = 23, indeed 23 x 7 = 161 = 1 mod 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 (unless you go to quantum computing!) ;
- 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-260 =
  221128255295296664352810852550262309276120895024
  700153944137483191288229414020019865127297265697
  465990859003300314000511707422045608592763579537
  571859542988389587092292384910067030341246205457
  845664136645406842143612930176940208
- 46391065875914794251435144458199

- 862 bits, or 260 decimal digits

RSA challenge, recent news

RSA-250 (829 bits)

2140324650240744961264423072839333563008614715144755017797754920
88141802344714013664334551909580467961099285187247091458768739
626192155736304745477052080511905649310668769159001975940569345
7452230589325976697471681738069364894699871578494975937497937 =

641352894770715802787901901705773890848250147429434472081168596
3202453234463 0238623598752668347708737661925585694639798853367

x

333720275949781565562260106053551142279407603447675546667845209
8023841729210037080257448673296881877565718986258036932062711

(> ~2700 CPU-core years, F. Boudot et al., Feb 2020)