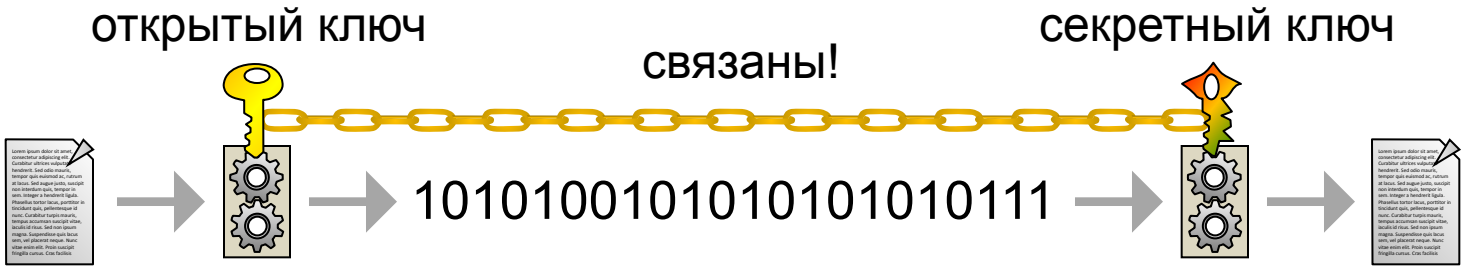
Информационная безопасность

§ 80. Современные алгоритмы шифрования

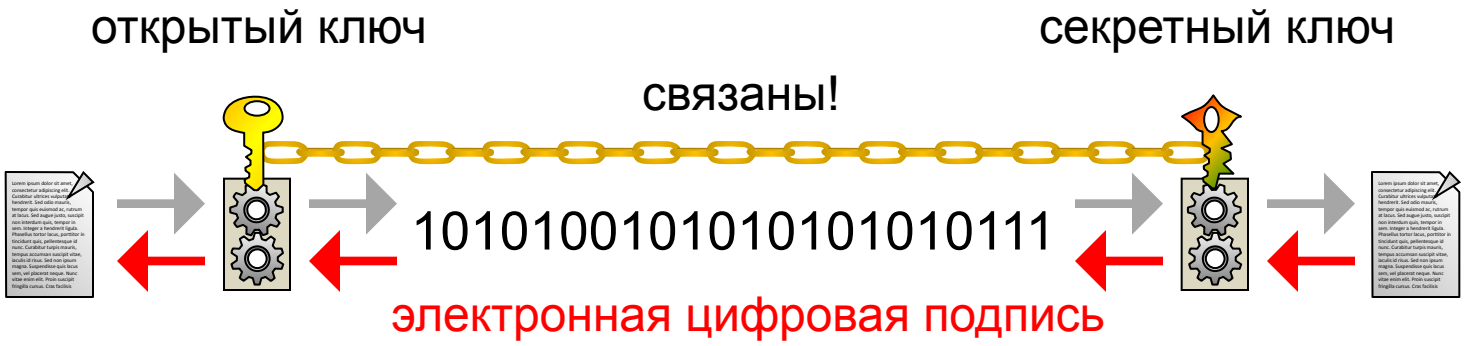
Алгоритм RSA

Р. Райвест (R. Rivest), А. Шамир (A. Shamir) и Л. Адлеман (L. Adleman), 1977.

Шифрование с открытым ключом:



Идея: применение открытого и секретного ключа восстанавливает сообщение:



Как построить ключи RSA?

1. Выбрать два **простых числа**, например,

$$p = 3, \quad q = 7$$

2. Вычислить

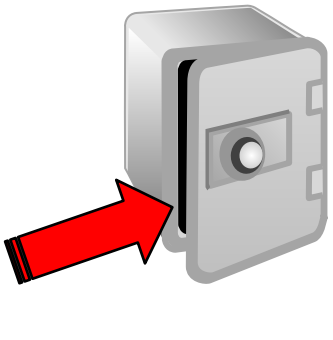
$$n = p \cdot q = 3 \cdot 7 = 21,$$
$$\varphi = (p-1) \cdot (q-1) = 2 \cdot 6 = 12$$

3. Выбрать число e ($1 < e < \phi$), которое не имеет общих делителей с ϕ : $e = 5$

4. Найти число d, для которого при некотором целом k выполняется условие: $d \cdot e = k \cdot \varphi + 1$

$$d = 17: \quad 17 \cdot 5 = 7 \cdot 12 + 1$$

- **Открытый ключ**: (e, n) $(5,21)$
- **Секретный ключ**: (d, n) $(17,21)$



Алгоритм RSA

Шифрование: открытый ключ (e, n)

1. Сообщение – последовательность чисел в интервале $[0, n-1]$.
2. Для каждого числа вычислить код

$$y = x^e \bmod n$$

Расшифровка: секретный ключ (d, n)

Для каждого кода вычислить число исходного сообщения:

$$x = y^d \bmod n$$

Алгоритм RSA: вычисление

Проблема:

очень большое число

$$y = x^e \bmod n$$

Упрощающая формула:

$$(a+b) \bmod n = (a \bmod n + b \bmod n) \bmod n$$

Доказательство:

$r_a = a \bmod n$ ь $r_b = b \bmod n$

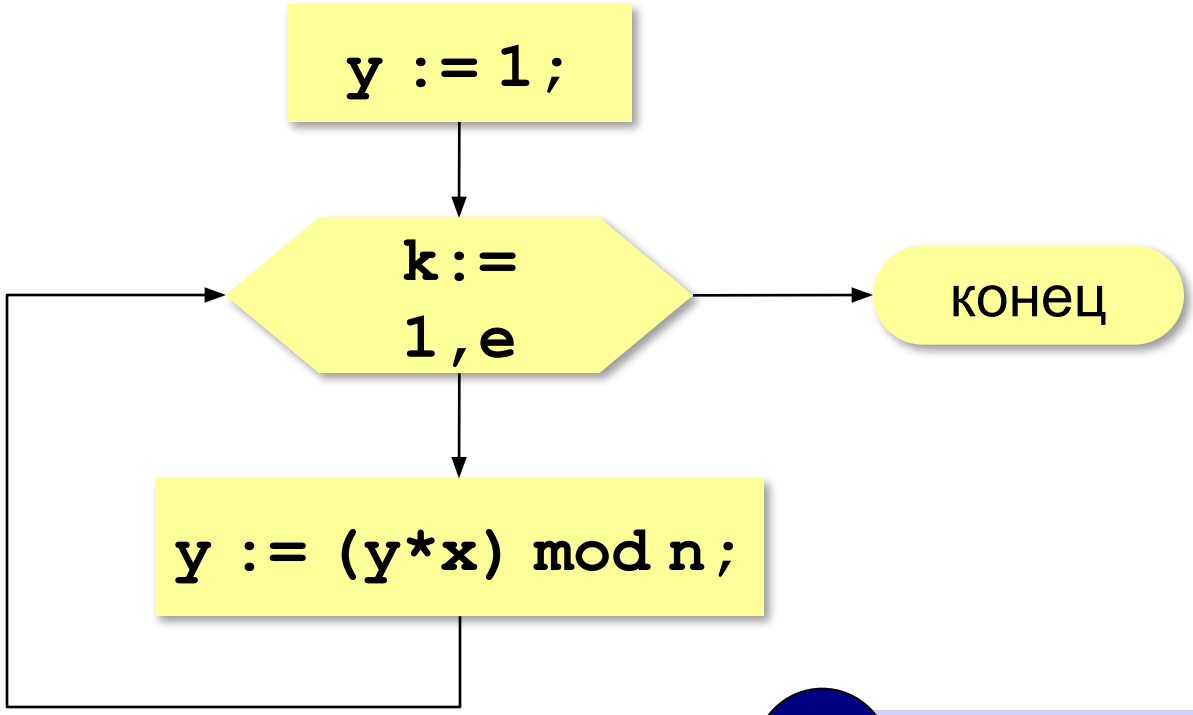
$$a = k \cdot n + r_a, \quad b = \square \cdot n + r_b$$

$$(a+b) \bmod n = \left[(k+\square) \cdot n + r_a + r_b\right] \bmod n$$

$$= (r_a + r_b) \bmod n$$

Алгоритм RSA: вычисление

Вычисление $y = x^e \bmod n$



? Как быстрее?

Быстрое возведение в степень

$$x^e = \begin{cases} x^{e/2} \cdot x^{e/2}, & e - \text{чётное} \\ x^{e-1} \cdot x, & e - \text{нечётное} \end{cases}$$

Пример:

$x^{100} = x^{64} \cdot x^{32} \cdot x^4$ — 2 умножения

$x^4 = [x^2]^2$ — 1 умножение

$x^{32} = [[[x^4]^2]^2]^2$ — 3 умножения

$x^{64} = [x^{32}]^2$ — 1 умножение

! Вместо 99!

Программирование: $x^e = [b]^k \cdot p$

$$x^7 = [x]^7 \cdot 1 = [x]^6 \cdot x = [x^2]^3 \cdot x = [x^2]^2 \cdot x^3$$
$$= [x^4]^1 \cdot x^3 = [x^4]^0 \cdot x^7 = x^7$$

Быстрое возведение в степень (+ mod)

```
def quickPowMod( x, e, n ):
  b, k, y = x, e, 1
  while k:
    if k % 2 == 0:
      k //= 2
      b = (b * b) % n
    else:
      k -= 1
      y = (b * y) % n
  return y
```

```
def powMod( x, e, n ):
  y = 1
  for k in range(e):
    y = (y*x) % n
  return y
```

28,5 сек

$$y = 123^{123456789} \bmod 1023$$

0,0000257 сек

Алгоритм RSA: пример

Сообщение: 1 2 3

Шифрование: открытый ключ (e,n) $(5,21)$

$$1 \Rightarrow 1^5 \bmod 21 = 1$$
$$2 \Rightarrow 2^5 \bmod 21 = 32 \bmod 21 = 11$$
$$3 \Rightarrow 3^5 \bmod 21 = 243 \bmod 21 = 12$$

зашифрованное сообщение: **1 11 12**

Расшифровка: секретный ключ (d,n) $(17,21)$

$$1 \Rightarrow 1^{17} \bmod 21 = 1$$
$$11 \Rightarrow 11^{17} \bmod 21 = 2$$
$$12 \Rightarrow 12^{17} \bmod 21 = 3$$

расшифрованное сообщение: **1 2 3**

Алгоритм RSA: вскрытие

Задача: при известном открытом ключе (e, n)
найти секретный ключ d

Способ:

1)разложить n на взаимно-простые множители:

$$n = p \cdot q$$

2)вычислить

$$\varphi = (p-1) \cdot (q-1)$$

3)найти d, такое что при некотором k

$$d \cdot e = k \cdot \varphi + 1$$

Проблема: разложение большого числа на простые множители требует недостижимого объема вычислений (при длине n > 1024 бита)

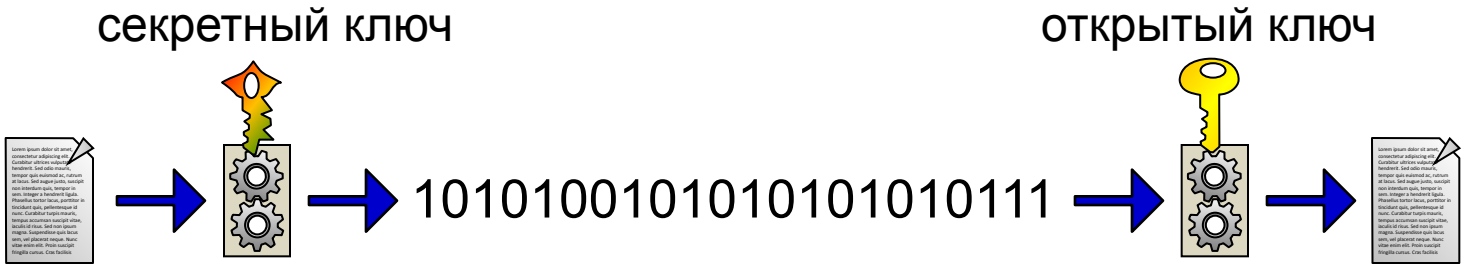
Алгоритм RSA

- для обмена открытыми ключами можно использовать незащищенный канал
- много готовых реализаций
- криптостойкость (при длине n > 1024 бита)

- медленная шифровка и (особенно) расшифровка
- при малом n взламывается

Электронная цифровая подпись

Электронная цифровая подпись (ЭЦП) – это набор символов, который получен в результате шифрования сообщения (или его хэш-кода) с помощью секретного ключа отправителя.



Применение:

- доказательство авторства
- невозможность отказа от авторства
- защита от изменений (проверка целостности)