ЛЕКЦИЯ 10.
Алгоритм шифрования RSA.

10.1. Структура алгоритма RSA.

10.2. Вычислительная реализация алгоритма RSA.

10.3. Криптоанализ алгоритма RSA.

Структура алгоритма RSA

Схема Райвеста-Шамира-Адлемана (RSA)
представляет собой

1. **Блочный шифр**, в котором и открытый текст, и шифрованный текст представляются *целыми* числами из диапазона от ***0*** до ***n*** **- 1** для некоторого ***n***. ***Длина блока*** должна быть меньше или равна $\log_2(n)$ **.**

2. На практике длина блока выбирается равной 2^k *битам*, где $2^k < n < 2^{k+1}$ **.**

3. Шифрование и дешифрование для блока открытого текста **M** и блока шифрованного текста **C** можно представить в виде следующих формул:

$$C = M^e \pmod{n},$$
$$M = C^d \pmod{n} = (M^e)^d \pmod{n} = M^{ed} \pmod{n}.$$

4. Как отправитель, так и получатель должны знать значение ***n***.

Отправитель знает значение *e*, и только **получателю** известно значение ***d****.*

Данная схема является алгоритмом шифрования
с **открытым ключом KU = {e, n}**, и ***личным*** ***ключом*** ***KR = {d, n}.***

Требования

к алгоритму шифрования с открытым ключом:

1.Должны существовать такие значения **e, d** и **n**, что

$$\mathbf{M^{ed} = M \ (mod\ n)} \text{ для всех } M<n .$$

2. Должны *относительно легко* вычисляться $\mathbf{M^e}$ и $\mathbf{C^d}$ для всех значений $M < n$..

3. Должно быть **практически невозможно** определить **d** по имеющимся **e** и **n**.

Анализ первого требования.

Необходимо найти соотношение вида:

$$\mathbf{M^{ed} = M \ (mod\ n).}$$

По *следствию из теоремы **Эйлера***: для таких любых двух простых чисел ***p*** и ***q*** и таких любых двух целых чисел ***n*** и ***m***, что ***n=pq*** и ***0<m<n***, и произвольного целого числа ***k*** выполняются соотношения:

$$\mathbf{m^{k\phi(n)+1} = m^{k(p-1)(q-1)+1} \equiv m \ (mod\ n),}$$

где ***ф(n)*** является функцией ***Эйлера***.

*В случае простых **p** и **q** имеем **ф(pq)*** = **(p - 1)(q - 1).**

Поэтому требуемое соотношение получается при условии

$$e\, d = k \times \phi(n) + 1.$$

Это эквивалентно следующим соотношениям:

$$e\, d \equiv 1 \ (mod\ \phi(n)),$$

$$d \equiv e^{-1} \ (mod\ \phi(n)),$$

т.е. ***e*** и ***d*** являются <u>взаимно обратными</u> по модулю ***ф(n).***

ОБЩИЙ АЛГОРИТМ ВЗАИМОДЕЙСТВИЯ
по схеме **RSA**

1. Пользователь ***A*** публикует свой открытый ключ.

2. Пользователь ***B*** собирается переслать ему сообщение **M** - пользователь ***B*** вычисляет шифрованное сообщение с помощью открытого ключа

$$C = M^e \pmod{n}$$

и пересылает шифр **C**.

3. Получив этот шифрованный текст, пользователь ***A*** дешифрует его, вычисляя с помощью личного ключа

$$M = C^d \pmod{n}.$$

Компоненты схемы RSA:

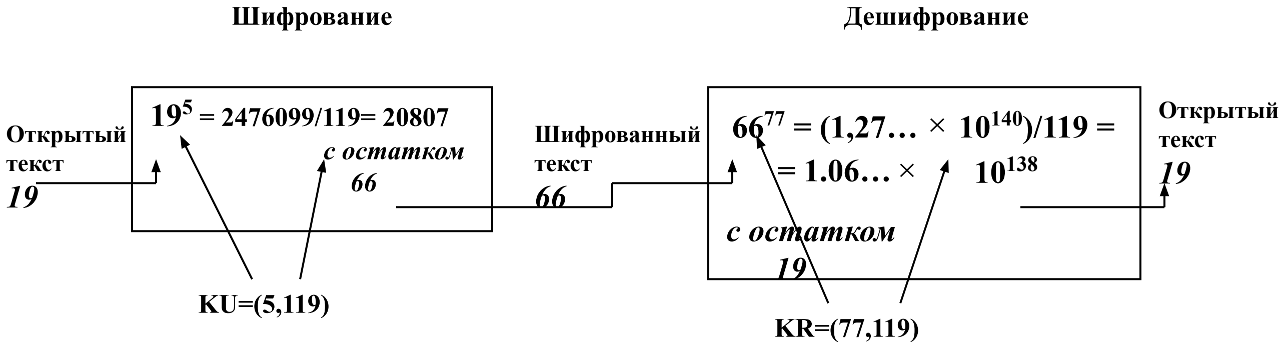
p и ***q*** — два простых числа (секретные, выбираются),
n* = *pq (***открытое***, вычисляется),
такое ***e***, что $(\phi(n), e) = 1$, $1 < e < \phi(n)$ (***открытое***, выбирается),
$d \equiv e^{-1} \pmod{\phi(n)}$ (**секретное**, вычисляется).

Личный ключ складывается из **{d, n}**, а **открытый** — из **{e, n}.**

Алгоритм RSA

Вычисление ключей	
Выбор ***p, q***	***p*** и ***q*** должны быть простыми
Вычисление $n = p \times q$	
Вычисление $\phi(n) = (p-1)(q-1)$	
Выбор целого ***e***	$(\phi(n), e) = 1$, $1 < e < \phi(n)$
Вычисление ***d***	$d \equiv e^{-1} \bmod \phi(n)$
Открытый ключ	$KU = \{e, n\}$
Личный ключ	$KR = \{d, n\}$
Шифрование	
Открытый текст:	$M<n$
Шифрованный текст:	$C = M^e \pmod n$
Дешифрование	
Открытый текст:	C
Дешифрованный текст:	$M = C^d \pmod n$

Пример реализации алгоритма RSA



В примере **ключи** вычисляются следующим образом:

1. Выбираются два простых числа: **p = 7** и ***q*** **= 17**.
2. Вычисляется ***n*** **= pq = 7** × ***17 = 119***
3. Вычисляется **ф(n) = (p - 1)(q - 1) =** ***96.***
4. Выбирается **e**, взаимно простое с **ф(n) = 96** и меньшее, чем **ф(n)**; в данном случае – **e = 5.**
5. Определяется такое **d**, что **de = 1 (mod 96)** и **d < 96**. Соответствующим значением будет **d = 77**, так как ***77*** × ***5 = 385 = 4*** × ***96 + 1.***
6. В результате получаются открытый ключ **KU = {5,119}** и личный ключ **KR={77,119}.**

Вычислительная реализация алгоритма RSA

Шифрование и дешифрование

Упрощение операции **возведения целого числа в целую степень** по ***модулю n:***

1. **[(a mod *n*) × (b mod *n*)] mod *n* = (a× b) mod *n*.**
2. В общем случае значение $a^x(\bmod n)$ вычисляется с помощью известной ***схемы Горнера***

«слева направо»

$$a^x(\bmod n) = (((a^{x_{k-1}})^2 \cdot a^{x_{k-2}})^2 \cdot \ldots \cdot a^{x_1})^2 \cdot a^{x_0}(\bmod n),$$

или «справа налево»

$$a^{x_0} \cdot (a^2)^{x_1} \cdot \ldots \cdot (a^{2^{k-1}})^{x_{k-1}} \bmod(n)$$

при *двоичном* разложении числа x в виде

$$x = \sum_{i=0}^{k-1} x_i 2^i$$

Учитывая, что ряд значений x_i при этом равен *нулю*, даже при больших числах x из интервала (*1, n*) вычисление всегда можно осуществить не более, чем за $O(\log_2(n))$ операций.

Вычисление ключей

Это означает выполнение следующих задач:

* определение двух простых чисел ***p*** и ***q***;
* выбор одного из чисел ***e*** или ***d*** и вычисление второго.

А) Обобщенная **процедура выбора простого числа**:

1. Выберите нечетное целое число ***n*** некоторым случайным образом (например, используя *генератор псевдослучайных чисел*).
2. Выберите целое число $a < n$ некоторым *случайным* образом.
3. Выполните *вероятностный тест* на простоту, например, ***тест Рабина***. Если ***n*** не выдерживает тестирования, отбросьте данное значение и перейдите к **п. 1**.

- Если ***n*** выдерживает достаточное число повторных тестов, примите данное значение ***n*** как подходящее, в противном случае перейдите к **п. 1**.

Б) Процесс вычисления ключей завершается выбором значения ***e*** и вычислением ***d*** *или, наоборот, выбором значения* ***d*** *и вычислением* ***e***.

В первом случае необходимо сначала выбрать такое ***e***, чтобы

$$(\phi(n), e) = 1,$$

потом вычислить

$$d \equiv e^{-1} \bmod \phi(n).$$

Обобщенный алгоритм Евклида

вычисляет наибольший общий делитель двух целых чисел и, если наибольший общий делитель оказывается равным 1, определяет обратное для одного из целых чисел по модулю другого (***мультипликативное обратное***).

1. генерирование случайных чисел,
2. сравнение их с $\phi(n)$ до тех пор, пока не будет найдено число, взаимно простое с $\phi(n)$.

При этом оказывается, что вероятность того, что два выбранных *случайно* числа окажутся ***взаимно простыми***, равна примерно **0,6**.

Криптоанализ алгоритма RSA

Три возможных подхода к криптоанализу алгоритма ***RSA***:

- **Простой перебор.** Предполагает проверку всех возможных личных ключей.
- **Математический анализ.** Подходы такого рода эквивалентны нахождению множителей произведения двух простых чисел.
- **Анализ временных затрат.** Опирается на анализ времени выполнения алгоритма дешифрования.

А) Защита против ***простого перебора*** в случае ***RSA*** — использование большого пространства ключей.

Б) Можно выделить **три** математически различных подхода к криптоанализу RSA.

- Разложение n на два его простых множителя. Это позволит вычислить $\phi(n)=(p-1)(q-1)$, на основании чего можно будет определить

$$d = e^{-1} \pmod{\phi(n)}.$$

- Определение непосредственно $\phi(n)$ без того, чтобы сначала определять p и q. Это также позволит определить

$$d = e^{-1} \pmod{\phi(n)}.$$

- Определение непосредственно d без того, чтобы сначала определять $\phi(n)$.

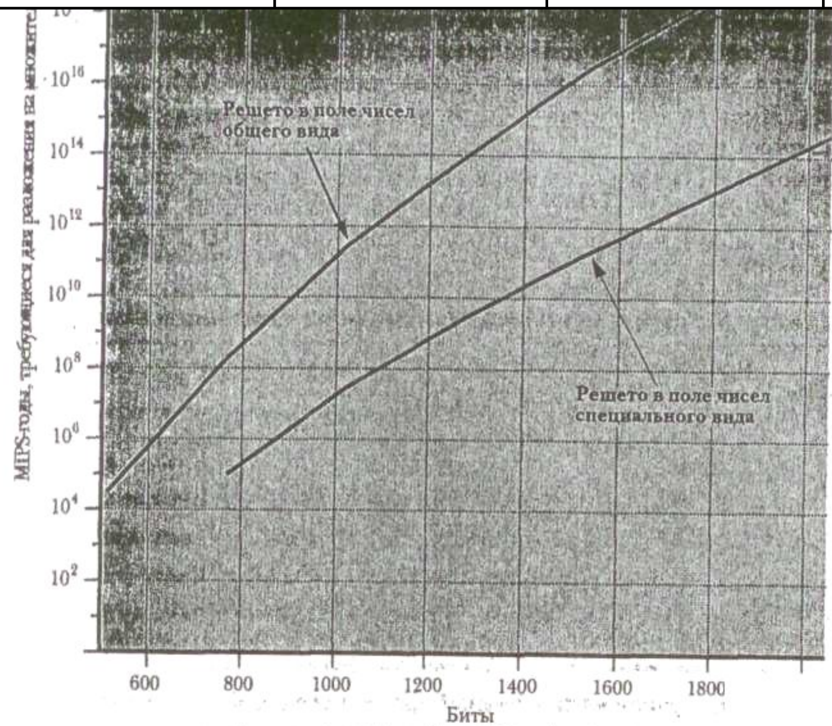
Задача определения $\phi(n)$ по данному n
оказывается эквивалентной **задаче разложения *n* на множители.**

Проблема определения d по данным e и n
оказывается требующей таких же затрат времени, как и
проблема разложения на множители.

Затраты на решение задачи разложения на множители
можно использовать в качестве *эталона* при оценке степени защищенности ***RSA***.

Прогресс в решении проблемы разложения на множители

Число десятичных знаков	Приблизительное число битов	Дата решения	Требуемое число MIPS-лет	Использованный алгоритм
100	332	Апрель 1991 г.	7	Квадратичное решето
110	365	Апрель 1992 г.	75	Квадратичное решето
120	398	Июнь 1993 г.	830	Квадратичное решето
129	428	Апрель 1994 г.	5000	Квадратичное решето
130	431	Апрель 1996 г.	500	Решето в поле чисел общего вида



MIPS-годы, требуемые для разложения на множители

Ограничения **относительно *p* и *q*:**

1. Значения ***p*** и ***q*** должны различаться по длине всего на несколько разрядов.

Например, и ***p***, и ***q*** должны попадать в диапазон **от 10^{75} до 10^{100}**.

2. Как **(*p* - 1), так и (*q* - 1),** должны содержать в своих разложениях достаточно *большой простой* множитель.

3. **((*p* – 1), (*q* - 1))** должен быть достаточно *малым*.

4. Показано, что если ***e* < *n*** и ***d* < ¼ *n*, то *d*** можно определить достаточно легко.

В) В данном случае противник получает возможность определить личный ключ, анализируя ***затраты времени***, которые требуются компьютеру для расшифровки сообщений.

Контрмеры против **анализа временных затрат**:

- **Постоянное время выполнения операции возведения в степень.** Изменение алгоритма таким образом, чтобы все возведения в степень занимали *одно и то же время* от начала выполнения до возврата результата. (при этом **увеличивается общее время** выполнения алгоритма).
- **Случайные задержки.** Меньшее влияние на общее время выполнения вызывает добавление в алгоритм возведения в степень *случайных задержек*, что уменьшает пользу от анализа временных затрат.
- **Маскировка.** Умножение шифрованного текста на *случайное число* перед тем, как выполнять возведение в степень.

Это не даст противнику возможности провести поразрядный анализ, *который является существенной частью подхода, основанного на анализе временных затрат.*

При использовании *функции маскировки* операция

$M = C^d$ (mod n) с личным ключом

выполняется следующим образом:

1. Генерируется ***секретное случайное*** число ***r*** в диапазоне от **0** до ***n-1***.
2. Вычисляется **$C' = C^{re}$ (mod n)** , где ***e*** является **открытым** значением показателя степени.
3. Вычисляется **$M' = (C')^d$ (mod n)** для обычной реализации ***RSA***.
4. Вычисляется **$M = M' r^{-1}$ (mod n),**

где **r^{-1}** - ***мультипликативное обратное значение r*** **(mod n).**

Аппаратные реализации RSA

Компания	Тактовая частота	Скорость передачи в Бодах на 512 бит	Тактовые циклы для шифрования 512 бит	Технология	Битов на микросхему	Количество транзисторов
Alpha Techn.	25 МГц	13К	0.98 М	2 микрона	1024	180000
AT&T	15 МГц	19К	0.4 М	1.5 микрона	298	100000
British Telecom	10 МГц	5.1К	1 М	2.5 микрона	256	----
Business Sim. Ltd.	5 МГц	3.8К	0.67 М	Вентильная матрица	32	----
CalmosSyst-Inc.	20 МГц	2.8К	0.36 М	2 микрона	593	95000
CNET	25 МГц	5.3К	2.3 М	1 микрон	1024	100000
Cryptech	14 МГц	17К.	0.4 М	Вентильная матрица	120	33000
Cylink	30 МГц	6.8К	1.2М	1.5 микрона	1024	150000
GEC Marconi	25 МГц	10.2К	0.67 М	1.4 микрона	512	160000
Pijnenburg	25 МГц	50К	0.256 М	1 микрон	1024	400000
Sandia	8 МГц	10К	0.4 М	2 микрона	272	86000
Siemens	5 МГц	8.5К	0.03 М	1 микрон	512	60000

Аппаратно *RSA* примерно в 1000 раз медленнее ***DES***.

Программно *DES* примерно в 100 раз быстрее ***RSA***.

Скорости программного шифрования RSA для различных длин модулей при 8-битовом открытом ключе

	512 битов	**768** битов	**1024** бита
Шифрование	0.03 с	0.05 с	0.08 с
Дешифрирование	0.16 с	0.48 с	0.93 с
Подпись	0.16 с	0.52 с	0.97 с
Проверка	0.02 с	0.07 с	0.08 с

*Шифрование **RSA** выполняется намного быстрее, если правильно выбрать значение **e**.*

Тремя наиболее частыми вариантами являются

3, 17 и **65537 (2^{16} + 1).**