Министерство цифрового развития, связи и массовых  
коммуникаций российской федерации

Государственное бюджетное образовательное учреждение

Высшего образования

Ордена Трудового Красного Знамени

МОСКОВСКИЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ СВЯЗИ И ИНФОРМАТИКИ

Курсовая работа

По теме:

“ Криптографические методы защиты информации”.

Выполнил студент

группы БАС 2101:

Кубышев Артём Сергеевич

Вариант 9

Проверил

Профессор кафедры ИБ Шелухин

Москва 2023

Cодержание:

[**Задание №1. Алгоритм шифрования ГОСТ 28147-89.** 4](#_Toc151393035)

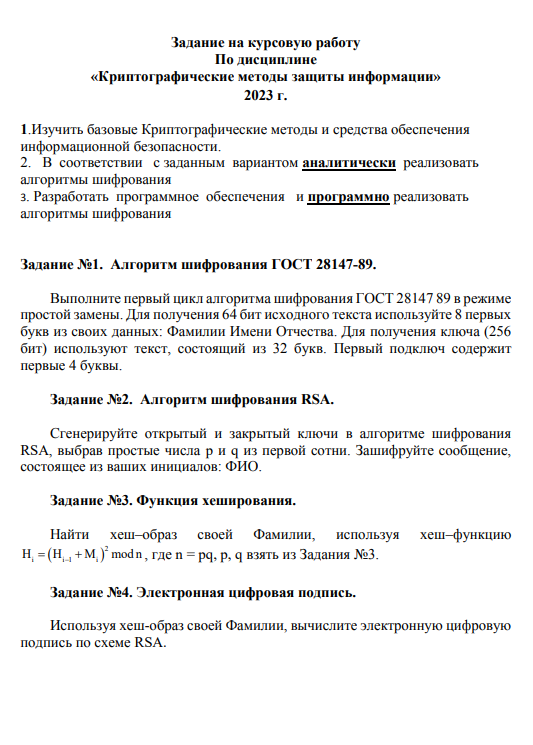
[**Задание №2. Алгоритм шифрования RSA.** I. Генерация ключей 7](#_Toc151393036)

[**Задание №3. Функция хеширования.** 10](#_Toc151393037)

[**Задание №4. Электронная цифровая подпись.** 12](#_Toc151393038)

[**Выводы:** 17](#_Toc151393039)

[**Приложения** 18](#_Toc151393040)



Задание №1. Алгоритм шифрования ГОСТ 28147-89.

Исходные данные для зашифрования: КУБЫШЕВА. Для ключа возьмем последовательность состоящую из 4 букв: СЕРГ**.** Для первого подключа Х используем первые 4 буквы ключа: СЕРГ. Переводим исходный текст и первый подключ в двоичную последовательность (см. Приложение Б):

Таблица 1- текст в двоичном виде.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | исходный текст | | первый подключ X0 | |
|  | Символ | код | Символ | код |
| 1 | К | 11001010 | С | 11010001 |
| 2 | У | 11010011 | Е | 11000101 |
| 3 | Б | 11000001 | Р | 11010000 |
| 4 | Ы | 11011011 | Г | 11000011 |
| 5 | Ш | 11011000 |  |  |
| 6 | Е | 11000101 |  |  |
| 7 | В | 11000010 |  |  |
| 8 | А | 11000000 |  |  |

входная последовательность  
L0: 11001010 11010011 11000001 11011011

R0: 11011000 11000101 11000010 11000000

следующие 32 бита определяют первый подключ  
Х0: 11010001 11000101 11010000 11000011

I. Найдем значение функции преобразования f(R0,X0) (см. Приложение А)  
1). Вычисление суммы R0 и X0 по mod 232

R0: 11011000 11000101 11000010 11000000

Х0: 11010001 11000101 11010000 11000011

X0+R0:

10101010 10001011 10010011 100000112

Результат суммирования (𝑅0+𝑋0) 𝑚𝑜𝑑 232: 1010 1010 1000 1011 1001 0011 1000 0011

Затем для каждого 4-битного блока вычислим его адрес в таблице подстановки. Номер блока соответствует номеру столбца, десятичное значение блока соответствует номеру строки в таблице.

Номера блоков 8 7 6 5 4 3 2 1

1010 1010 1000 1011 1001 0011 1000 0011 соответствующие номера строк в таблице подстановки

Таблица 2 – Таблица перестановки.

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

Таблица 3 – номер блоков.

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Столбец | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 |
| Группа | 1010 | 1010 | 1000 | 1011 | 1001 | 0011 | 1000 | 0011 |
| Строка | 10 | 10 | 8 | 11 | 9 | 3 | 8 | 3 |
| Заполнение | 3 | 14 | 3 | 14 | 4 | 13 | 2 | 2 |
| **Результат** | 0011 | 1110 | 0011 | 1110 | 0100 | 1101 | 0010 | 0010 |

Таблица 4 - Циклический сдвиг на 11 бит влево.

|  |  |
| --- | --- |
| Исходный | **0011 1110 001**1 1110 0100 1101 0010 0010 |
| Сдвинутый | 1111 0010 0110 1001 0001 0**001 1111 0001** |

f(R0,X0) = 1111 0010 0110 1001 0001 0001 1111 0001

**Вычисляем R1= f(R0,X0) ⊕L0.**

Результат преобразования функции f(R0,X0) складываем с L0 по mod2:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| L0: | 11001000 | 11000010 | 11000000 | 11001101 |
| f: | 11110010 | 01101001 | 00010001 | 11110001 |
| R1 | 00111000 | 10111011 | 11010000 | 00101010 |

L1=R0=11011000 11000101 11000010 11000000

Ответ:

L1: 11011000 11000101 11000010 11000000

R1: 00111000 10111011 11010000 00101010

Задание №2. Алгоритм шифрования RSA.I. Генерация ключей

Выберем два простых числа ***р* =41 и *q* =29**

Тогда модуль *n* = *pq*=41\*29 =1189 и функция Эйлера ϕ*(n)* = *(p-1)(q-1)* = 20\*28 = 1120

Закрытый ключ *d* выбираем из условий *d* < ϕ*(n)* и *d* взаимно просто с ϕ*(n)*, т.е. *d* и ϕ*(n)* не имеют общих делителей.  
Пусть *d* = 3.

Открытый ключ *e* выбираем из условий *e*<ϕ*(n)* и *de*=1*(mod* ϕ*(n)): e*<1120,

3\**e*=1(mod 1120).  
Последнее условие означает, что число 11\**e*-1 должно делиться на 1120 без остатка.  
Таким образом, для определения *e* нужно подобрать такое число *k*, что 3\**e*-1 = 1120\**k*.  
При *k*=2 получаем 3\**e*=2240+1

*e*=747.  
В нашем примере  
(747, 1189) – открытый ключ,  
(3, 1189) – секретный ключ

II**.** Шифрование.

Представим шифруемое сообщение **“КАС”** как последовательность целых чисел, в соответствии с русским алфавитом.

Таблица 5 – Последовательность букв русского алфавита.

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Буква | А | Б | В | Г | Д | Е | Ё | Ж | З | И | Й | К | Л | М | Н | О | П |
| Код | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 | 16 | 17 |
| Буква | Р | С | Т | У | Ф | Х | Ц | Ч | Ш | Щ | Ъ | Ы | Ь | Э | Ю | Я |  |
| Код | 18 | 19 | 20 | 21 | 22 | 23 | 24 | 25 | 26 | 27 | 28 | 29 | 30 | 31 | 32 | 33 |  |

Тогда, сообщение примет следующий вид:

Формула для шифрования: .

Зашифруем сообщение, используя открытый ключ (747, 1189):

1

Таким образом, исходному сообщению (12, 1, 19) соответствует криптограмма

(1003, 1, 298).

III. РасшифрованиеРасшифруем сообщение (1003, 1, 298), пользуясь секретным ключом (3,1189):

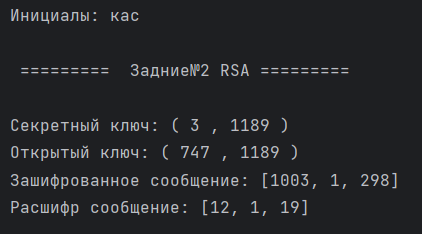
Расшифрование производится по формуле:

В результате расшифрования было получено исходное сообщение (12, 1, 19), то есть "КАС"

Код:

import math  
  
def phi(p: int, q: int) :  
 return (p - 1) \* (q - 1)  
  
def make\_table(alphabet: str) -> dict:  
 table: dict = {}  
 for i, char in enumerate(alphabet):  
 table[char] = i + 1  
 return table  
  
  
def get\_M(word: str) -> list[int]:  
 M: list[int] = list()  
 for char in word:  
 M.append(table[char])  
 return M  
  
def get\_d(phi: int, d) :  
 while (d < phi):  
 if (math.gcd(d, phi) == 1):  
 break  
 else:  
 d += 1  
 return d  
  
  
def get\_e(d: int, phi: int) :  
 tmp: list[int] = []  
 for k in range(1, d):  
 if (phi \* k + 1) % d == 0:  
 e = int((phi \* k + 1) / d)  
 return e  
  
  
# p и q поставить по заданию №2  
p = 41  
q = 29  
  
N= p \* q  
  
alphabet: str = "абвгдеёжзийклмнопрстуфхцчшщъыьэюя"  
  
  
phi = phi(p, q)  
  
  
table: dict = make\_table(alphabet)  
  
  
fio: str = "кубышев артём сергеевич"  
  
slovo: str = fio.split(" ")[0]  
  
initials = "".join([s[0] for s in fio.split(" ")])  
print("Инициалы:", initials)  
  
  
print("\n", "===" \* 3, " Задние№2 RSA", "===" \* 3, "\n")  
  
d=2  
d: int = get\_d(phi,d)  
  
  
e: int = get\_e(d, phi)  
  
  
M: list[int] = get\_M(initials)  
  
C: list[int] = []  
for i in range(len(initials)):  
 C.append((M[i] \*\* e) % N)  
  
  
D: list[int] = []  
for c in C:  
 D.append((c \*\* d) % N)  
  
  
print("Секретный ключ:","(", d,",",N,")")  
print("Открытый ключ:", "(",e,",",N,")")  
print("Зашифрованное сообщение:", C)  
print("Расшифр сообщение:", D)

Вывод программы:



Задание №3. Функция хеширования.

Хэшируемое сообщение «КУБЫШЕВ».  
Возьмем два простых числа p=41, q=29

Определим n=pq=1189.

Формула для хеширования:

Слово «КУБЫШЕВ.» можно представить последовательностью чисел (12, 21, 2, 29, 26, 6, 3) по номерам букв в алфавите.

Таблица 6 – Последовательность букв русского алфавита.

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Буква | А | Б | В | Г | Д | Е | Ё | Ж | З | И | Й | К | Л | М | Н | О | П |
| Код | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 | 16 | 17 |
| Буква | Р | С | Т | У | Ф | Х | Ц | Ч | Ш | Щ | Ъ | Ы | Ь | Э | Ю | Я |  |
| Код | 18 | 19 | 20 | 21 | 22 | 23 | 24 | 25 | 26 | 27 | 28 | 29 | 30 | 31 | 32 | 33 |  |

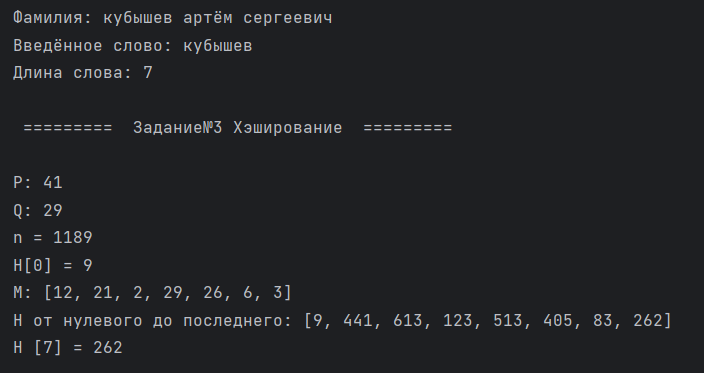
441mod 1189 =441

В итоге получаем хеш-образ сообщения «КУБЫШЕВ», равный 262

Код:

def formula(M: list[int], h0: int, n: int) -> list[int]:  
 h = [h0]  
 for mi in M:  
 h.append(((h[-1] + mi) \*\* 2) % n)  
 return h  
  
  
def make\_table(alphabet: str) -> dict:  
 table: dict = {}  
 for i, char in enumerate(alphabet):  
 table[char] = i + 1  
 return table  
  
  
def get\_M(word: str) -> list[int]:  
 M: list[int] = list()  
 for char in word:  
 M.append(table[char])  
 return M  
  
  
  
  
# p и q выбрать по варианту для Хэширования  
p: int = 41  
q: int = 29  
  
N: int = p \* q  
  
h0: int = 9  
  
  
alphabet: str = "абвгдеёжзийклмнопрстуфхцчшщъыьэюя"  
  
table: dict = make\_table(alphabet)  
  
  
surname: str = "кубышев артём сергеевич"  
  
sur: str = surname.split(" ")[0]  
  
print("Фамилия:", surname)  
print(f"Введённое слово: {sur}")  
print(f"Длина слова: {len(sur)}")  
  
print("\n", "===" \* 3, " Задание№3 Хэширование ", "===" \* 3, "\n")  
  
  
M: list[int] = get\_M(sur)  
  
h: int = formula(M, h0, N)  
  
H: int = h[-1]  
  
print("P:", p)  
print("Q:", q)  
print("n =", N)  
print("H[0] =", h0)  
print("M:", M)  
print("H от нулевого до последнего:", h)  
print("H",[len(sur)],"=",H)

Вывод программы:



**Задание №4. Электронная цифровая подпись.**

, ,

Найдём хэш:

Хэшируемое сообщение «КУБЫШЕВ».

Формула для хеширования:

Слово «КУБЫШЕВ.» можно представить последовательностью чисел (12, 21, 2, 29, 26, 6, 3) по номерам букв в алфавите.

441mod 1927=441

В итоге получаем хеш-образ сообщения «КУБЫШЕВ», равный 303

Отсюда, .

**Ключи шифрования RSA**

Тогда модуль *n* = *pq*= 1927 и функция Эйлера ϕ*(n)* = *(p-1)(q-1)*  = 1840

Закрытый ключ *d* выбираем из условий *d* < ϕ*(n)* и *d* взаимно просто с ϕ*(n)*, т.е. *d* и ϕ*(n)* не имеют общих делителей.  
Пусть *d* = 7.   
Открытый ключ *e* выбираем из условий *e*<ϕ*(n)* и *de*=1*(mod* ϕ*(n)): e*<1840,

7\**e*=1(mod 1840).  
Последнее условие означает, что число 7\**e*-1 должно делиться на 1840 без остатка.  
Таким образом, для определения *e* нужно подобрать такое число *k*, что 7\**e*-1 = 1840\**k*.  
При *k*=1 получаем 7\**e*=1840+1

*e*=263.  
В нашем примере  
(263, 1927) – открытый ключ,  
(7, 1927) – секретный ключ

**Подпись**

Тогда электронная цифровая подпись будет вычислена по формуле (при условии, что известен закрытый ключ):

Тогда, по формуле:

**Проверка**

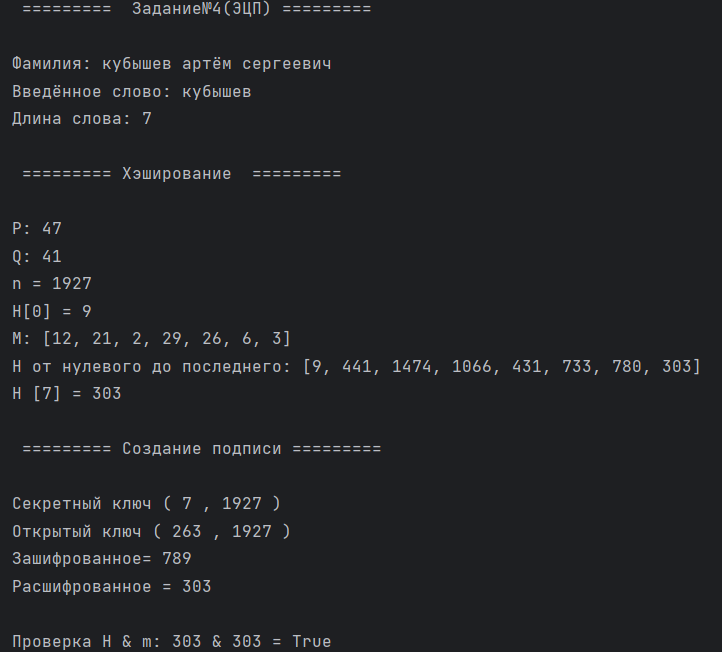
Проверка осуществляется по формуле (при условии, что известен открытый ключ):

Сравнив исходный хэш, и хэш полученный из формулы выше, делаем вывод о подлинности подписи: если совпало – то подпись подлинная, если нет – подделанная.

Проверим подпись: . Из равенства можно сделать вывод, что подпись – подлинная.

Код:

import math  
  
def phi(p: int, q: int) :  
 return (p - 1) \* (q - 1)  
  
  
  
def get\_d(phi: int, d) :  
 while (d < phi):  
 if (math.gcd(d, phi) == 1):  
 break  
 else:  
 d += 1  
 return d  
def get\_e(d: int, phi: int) :  
 tmp: list[int] = []  
 for k in range(1, d):  
 if (phi \* k + 1) % d == 0:  
 e = int((phi \* k + 1) / d)  
 return e  
  
def formula(M: list[int], h0: int, n: int) -> list[int]:  
 h = [h0]  
 for mi in M:  
 h.append(((h[-1] + mi) \*\* 2) % n)  
 return h  
  
  
def make\_table(alphabet: str) -> dict:  
 table: dict = {}  
 for i, char in enumerate(alphabet):  
 table[char] = i + 1  
 return table  
  
  
def get\_M(word: str) -> list[int]:  
 M: list[int] = list()  
 for char in word:  
 M.append(table[char])  
 return M  
  
  
  
  
  
print("\n", "===" \* 3, " Задание№4(ЭЦП)", "===" \* 3, "\n")  
  
#Поставить p и q по своему варианту, и H из 3-го задания  
p: int = 47  
q: int = 41  
N: int = p \* q  
h0: int = 9  
  
  
alphabet: str = "абвгдеёжзийклмнопрстуфхцчшщъыьэюя"  
  
table: dict = make\_table(alphabet)  
  
  
surname: str = "кубышев артём сергеевич"  
  
sur: str = surname.split(" ")[0]  
  
print("Фамилия:", surname)  
print(f"Введённое слово: {sur}")  
print(f"Длина слова: {len(sur)}")  
  
print("\n", "===" \* 3, "Хэширование ", "===" \* 3, "\n")  
  
  
M: list[int] = get\_M(sur)  
  
h: int = formula(M, h0, N)  
  
H: int = h[-1]  
  
print("P:", p)  
print("Q:", q)  
print("n =", N)  
print("H[0] =", h0)  
print("M:", M)  
print("H от нулевого до последнего:", h)  
print("H",[len(sur)],"=",H)  
  
  
fin: int = phi(p, q)  
  
# генерация новых Д и Е  
d=5  
d: int = get\_d(fin,d)  
  
e: int = get\_e(d, fin)  
  
S: int = (H \*\* d) % N  
  
m: int = (S \*\* e) % N  
  
  
print("\n", "===" \* 3, "Создание подписи", "===" \* 3, "\n")  
print("Секретный ключ","(", d,",",N,")")  
print("Открытый ключ","(", e,",",N,")")  
print("Зашифрованное=", S)  
print("Расшифрованное =", m)  
  
  
print()  
print(f"Проверка H & m: {H} & {m} = {H == m}")

Вывод программы:

**Выводы:**

В данной курсовой работе я использовал полученные знания на основе лабораторных работ. Благодаря данной работе я научился использовать новые методы шифрования.

**Приложения**

#### Приложение А. Алгоритм шифрования ГОСТ 28147-89

Межгосударственный стандарт шифрования ГОСТ 28147-89 предусматривает 4 режима работы:

* режим простой замены;
* режим гаммирования;
* режим гаммирования с обратной связью;
* режим выработки имитовставки.

#### Простая замена.

Режим простой замены является основой для всех остальных режимов. Длина блока - 64 бита, длина ключа – 256 бит, количество подключей – 32, длина подключа - 32 бита, число циклов –32.

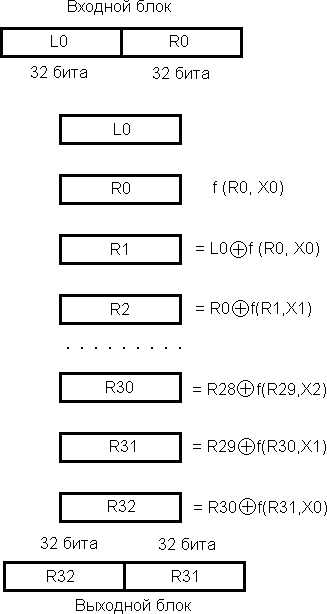
Открытые данные, подлежащие зашифрованию, разбиваются на 64-битные блоки, которые обрабатываются независимо друг от друга (Так как блоки данных шифруются независимо друг от друга, при зашифровании двух одинаковых блоков открытого текста получаются одинаковые блоки шифротекста и наоборот.). Схема обработки 64-битного блока показана на Рис.1- 2.

Процедура зашифрования 64-битного блока включает 32 цикла. В каждом цикле используется свой подключ, который вырабатывается из основного ключа. Размер массива открытых или зашифрованных данных, подвергающийся соответственно зашифрованию или расшифрованию, должен быть кратен 64 битам, после выполнения операции размер полученного массива данных не изменяется.

Режим простой замены применяется для шифрования короткой, ключевой информации.

В режимах гаммирования вырабатывается гамма шифра блоками по 64 бита с применением ГОСТ в режиме простой замены. В первом режиме гамма не зависит от шифруемых данных, во втором – зависит от шифрблоков.

Режим выработки имитовставки предназначен для обнаружения случайных или умышленных искажений данных. Имитовставка вырабатывается (с помощью первых 16 циклов ГОСТ в режиме простой замены) из открытых данных и ключа и добавляется при передаче по каналу связи к блокам зашифрованных данных.



где Å - сложение по модулю 2

Рис. 1. Алгоритм шифрования ГОСТ 28147-89 (режим простой замены).

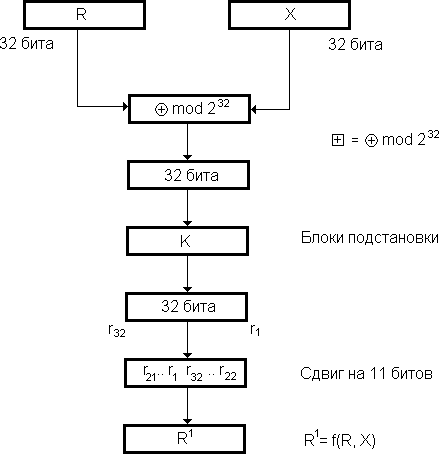


Рис. 2. Функция преобразования f(R,X) в алгоритме ГОСТ 28147-89.

Приложение Б. Символы кириллицы (альтернативная кодовая таблица ASCII)

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Сим-л | Дес  . | Двоич. | Сим-л | Дес. | Двоич. |
| А | 192 | 11000000 | б | 225 | 11100001 |
| Б | 193 | 11000001 | в | 226 | 11100010 |
| В | 194 | 11000010 | г | 227 | 11100011 |
| Г | 195 | 11000011 | д | 228 | 11100100 |
| Д | 196 | 11000100 | е | 229 | 11100101 |
| Е | 197 | 11000101 | ж | 230 | 11100110 |
| Ж | 198 | 11000110 | з | 231 | 11100111 |
| З | 199 | 11000111 | и | 232 | 11101000 |
| И | 200 | 11001000 | й | 277 | 11101001 |
| Й | 201 | 11001001 | к | 234 | 11101010 |
| К | 202 | 11001010 | л | 235 | 11101011 |
| Л | 203 | 11001011 | м | 236 | 11101100 |
| М | 204 | 11001100 | н | 237 | 11101101 |
| Н | 205 | 11001101 | о | 238 | 11101110 |
| О | 206 | 11001110 | п | 239 | 11101111 |
| П | 207 | 11001111 | р | 240 | 11110000 |
| Р | 208 | 11010000 | с | 241 | 11110001 |
| С | 209 | 11010001 | т | 242 | 11110010 |
| Т | 210 | 11010010 | у | 243 | 11110011 |
| У | 211 | 11010011 | ф | 244 | 11110100 |
| Ф | 212 | 11010100 | х | 245 | 11110101 |
| Х | 213 | 11010101 | ц | 246 | 11110110 |
| Ц | 214 | 11010110 | ч | 247 | 11110111 |
| Ч | 215 | 11010111 | ш | 248 | 11111000 |
| Ш | 216 | 11011000 | щ | 249 | 11111001 |
| Щ | 217 | 11011001 | ъ | 250 | 11111010 |
| Ъ | 218 | 11011010 | ы | 251 | 11111011 |
| Ы | 219 | 11011011 | ь | 252 | 11111100 |
| Ь | 220 | 11011100 | э | 253 | 11111101 |
| Э | 221 | 11011101 | ю | 254 | 11111110 |
| Ю | 222 | 11011110 | я | 255 | 11111111 |
| Я | 223 | 11011111 | пробел | 32 | 00010000 |
| а | 224 | 11100000 |  |  |  |

Приложение В. Блок подстановки в алгоритме шифрования ГОСТ 28147-89

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

Пример. Пусть 32-битная последовательность имеет вид

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 100  1 | 101  1 | 110  0 | 010  1 | 111  0 | 010  0 | 000  0 | 100  1 |

Разобьем входную последовательность на 8 блоков по 4 бита. Шестой блок 1100 пропускаем через 6-ой узел подстановки по следующему правилу: преобразуем двоичное число 1100 к десятичному виду – 12. Заполнение 12-ой строки для 6-ого узла подстановки равно 9, что в двоичном виде есть 1001. Таким образом, 4-битный блок 1100 заменяется на 1001. Остальные блоки заменяются аналогично.

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | номер  узла |
| 100  1 | 101  1 | 110  0 | 010  1 | 111  0 | 010  0 | 000  0 | 100  1 | вход |
| 9 | 11 | 12 | 5 | 14 | 4 | 0 | 9 | адрес |
| 2 | 7 | 9 | 15 | 5 | 10 | 14 | 11 | заполне  ние |
| 001  0 | 011  1 | 100  1 | 111  1 | 010  1 | 101  0 | 111  0 | 101  1 | результа  т |

Выходная последовательность имеет вид

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 001  0 | 011  1 | 100  1 | 111  1 | 010  1 | 101  0 | 111  0 | 101  1 |

#### Приложение Г. Алгоритм шифрования RSA

Алгоритм шифрования RSA относится к криптографическим системам с открытым ключом. Криптосистемы с открытым ключом (асимметричные криптосистемы) были разработаны во второй половине семидесятых годов. В асимметричных криптосистемах процедуры прямого и обратного криптопреобразования выполняются на различных ключах и не имеют между собой очевидных и легко прослеживаемых связей, позволяющих по одному ключу определить другой. В такой схеме знание только ключа зашифрования не позволяет расшифровать сообщение, поэтому он не является секретным элементом шифра и обычно публикуется участником обмена для того, чтобы любой желающий мог послать ему шифрованное сообщение.

Принцип функционирования асимметричной криптосистемы заключается в следующем:

* пользователь А генерирует два ключа - открытый (незасекречен-ный) и секретный

- и передает открытый ключ по незащищенному каналу пользователю Б;

* пользователь Б шифрует сообщение, используя открытый ключ шифрования пользователя А;
* пользователь Б посылает зашифрованное сообщение пользователю А по незащищенному каналу;
* пользователь А получает зашифрованное сообщение и дешифрует его, используя свой секретный ключ.

Пары {открытый ключ; секретный ключ} вычисляются с помощью специальных алгоритмов, причем ни один ключ не может быть выведен из другого.

#### Криптографическая система RSA (Rivest-Shamir-Adleman)

Авторами алгоритма RSA, предложенного в 1977 г., являются Р.Риверст (Rivest), А.Шамир (Shamir) и А.Адлеман (Adleman). Надежность алгоритма основывается на трудности факторизации (разложения на множители) больших чисел и трудности вычисления дискретных алгоритмов (нахождения x при известных a, b и n из уравнения ax = b (mod n) ).

Алгоритм RSA состоит из трех частей: генерации ключей, шифрования и расшифрования.

1. Генерация ключей.

Выберем два больших различных простых числа p и q (Натуральное число называется простым, если оно делится только на себя и на 1.) и найдем их произведение

n = pq .

Вычислим функцию Эйлера j(n) по формуле

j(n) = (p-1)(q-1).

Закрытый ключ d выбираем из условий

d < j(n) и d взаимно просто с j(n), т.е. d и j(n) не имеют общих делителей.

Открытый ключ e выбираем из условий

e < j(n) и

de = 1(mod j(n)) .

Последнее условие означает, что разность de - 1 должна делить-ся на j(n) без остатка. Для определения числа e нужно подобрать такое число k, что

de - 1 = j(n)\*k .

В алгоритме RSA

( e, n ) – открытый ключ, ( d, n ) – секретный ключ.

1. Шифрование.

Исходное сообщение разбивается на блоки Mi одинаковой длины. Каждый блок представляется в виде большого десятичного числа, меньшего n, и шифруется отдельно. Шифрование блока M (M - десятичное число) осуществляется по следующей формуле

##### Me = C (mod n) ,

где C – шифрблок, соответствующий блоку открытого сообщения M. Шифрблоки соединяются в шифрограмму.

1. Расшифрование.

При расшифровании шифрограмма разбивается на блоки известной длины и каждый шифрблок расшифровывается отдельно по следующей формуле

Cd = M (mod n) .

#### Приложение Д. Таблица простых чисел

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 1 | 2 | 3 | 5 | 7 |
| 11 | 13 | 17 | 19 | 23 |
| 29 | 31 | 37 | 41 | 43 |
| 47 | 53 | 59 | 61 | 67 |
| 71 | 73 | 79 | 83 | 89 |
| 97 | 101 | 103 | 107 | 109 |
| 113 | 127 | 131 | 137 | 139 |
| 149 | 151 | 157 | 163 | 167 |
| 173 | 179 | 181 | 191 | 193 |
| 197 | 199 | 211 | 223 | 227 |
| 229 | 233 | 239 | 241 | 251 |
| 257 | 263 | 269 | 271 | 277 |
| 281 | 283 | 293 | 307 | 311 |
| 313 | 317 | 331 | 337 | 347 |
| 349 | 353 | 359 | 367 | 373 |
| 379 | 383 | 389 | 397 | 401 |
| 409 | 419 | 421 | 431 | 433 |
| 439 | 443 | 449 | 457 | 461 |
| 463 | 467 | 479 | 487 | 491 |
| 499 | 503 | 509 | 521 | 523 |
| 541 | 547 | 557 | 563 | 569 |
| 571 | 577 | 587 | 593 | 599 |

#### Приложение Е. Функция хеширования

Функцией хеширования (хеш-функцией) называется преобразо-вание данных, переводящее строку битов M произвольной длины в строку битов h(M) некоторой фиксированной длины (несколько десятков или сотен бит).

Хеш-функция h(M) должна удовлетворять следующим условиям:

1. хеш-функция h(M) должна быть чувствительна к любым изменениям входной последовательности M;
2. для данного значения h(M) должно быть невозможно найти значение M;
3. для данного значения h(M) должно быть невозможно найти значение M’ ¹ M

такое, что h(M‘) = h(M) .

Ситуация, при которой для различных входных последова-тельностей M , M‘

совпадают значения их хеш-образов: h(M) = h(M‘), называется коллизией.

При построении хеш-образа входная последовательность M разбивается на блоки Mi фиксированной длины и обрабатывается поблочно по формуле

Hi = f(Hi-1 , Mi).

Хеш-значение, вычисляемое при вводе последнего блока сообщения, становится хеш-значением (хеш-образом) всего сообщения.

В качестве примера рассмотрим упрощенный вариант хеш-функции из рекомендаций МККТТ Х.509:

Hi = (Hi-1 + Mi )2 mod n ,

где n = pq, p и q – большие простые числа, H0 - произвольное начальное заполнение,

Mi - i-тый блок сообщения M = M1 M2 … Mk .

#### Приложение Ж. Электронная цифровая подпись

Цифровая подпись в цифровых документах играет ту же роль, что и подпись, поставленная от руки в документах на бумаге: это данные, присоединяемые к передаваемому сообщению, подтвержда-ющие, что владелец подписи составил или заверил это сообщение. Получатель сообщения с помощью цифровой подписи может проверить, что автором сообщения является именно владелец подписи и что в процессе передачи не была нарушена целостность полученных данных.

При разработке механизма цифровой подписи возникают следующие задачи:

* создать подпись таким образом, чтобы ее невозможно было подделать;
* иметь возможность проверки того, что подпись действительно принадлежит указанному владельцу;
* иметь возможность предотвратить отказ от подписи.

#### Классическая схема создания цифровой подписи

При создании цифровой подписи по классической схеме отправитель

1. применяет к исходному сообщению хеш-функцию;
2. вычисляет цифровую подпись по хеш-образу сообщения с использованием секретного ключа создания подписи;
3. формирует новое сообщение, состоящее из исходного сообщения и добавленной к нему цифровой подписи.

Получатель, получив подписанное сообщение,

1. отделяет цифровую подпись от основного сообщения;
2. применяет к основному сообщению хеш-функцию;
3. с использованием открытого ключа проверки подписи извлекает хеш-образ сообщения из цифровой подписи;
4. проверяет соответствие вычисленного хеш-образа сообщения (п.2) и извлеченного из цифровой подписи. Если хеш-образы совпадают, то подпись признается подлинной.

#### Схема подписи RSA

Криптосистема с открытым ключом RSA может использоваться не только для шифрования, но и для построения схемы цифровой подписи.

Для создания подписи сообщения M отправитель

1. вычисляет хеш-образ r = h(M) сообщения M с помощью некоторой хеш- функции;
2. зашифровывает полученный хеш-образ r на своем секретном ключе (d,n) , т.е. вычисляет значение s = rd mod n , которое и является подписью.

Для проверки подписи получатель

1. расшифровывает подпись s на открытом ключе (e,n) отправителя, т.е. вычисляет r’ = se mod n и таким образом восстанавливает предполагаемый хеш-образ r’ сообщения M;
2. вычисляет хеш-образ h(M) = r сообщения M с помощью той же самой хеш- функции, которую использовал отправитель;
3. сравнивает полученные значения r и r’ . Если они совпадают, то подпись правильная, отправитель действительно является тем, за кого себя выдает, и сообщение не было изменено при передаче.