Министерство образования Республики Беларусь

Учреждение образования «Белорусский государственный университет   
информатики и радиоэлектроники»

Факультет компьютерных систем и сетей

Кафедра информатики

Дисциплина: Методы Защиты Информации

ОТЧЁТ

к лабораторной работе №3

на тему

**Асимметричная криптография. Криптосистема Рабина**

Выполнил: студент гр. 253505

Сенько Н. С.

Проверил: ассистент кафедры

информатики

Герчик А. В.

Минск 2025

**СОДЕРЖАНИЕ**

[1 Цель работы 3](#_Toc207722280)

[2 Ход работы 4](#_Toc207722281)

[Заключение](#_Toc207722282) 6

[Приложение А (обязательное) исходный код программного продукта](#_Toc207722283) 7

# 1 ЦЕЛЬ РАБОТЫ

Целью данной лабораторной работы является изучение принципов асимметричная шифрования на примере криптосистемы Рабина.

В рамках работы требуется разработать программные средство для шифрования и дешифрования данных, закрепить навыки работы с асимметричными алгоритмами шифрования, освоить процесс преобразования данных с использованием открытых и закрытых ключей.

Результатом выполнения работы должен быть скрипт, который позволяет:

– зашифровать входные данные используя криптосистему Рабина.

– расшифровать зашифрованные данные обратно в набор из четырех возможных значений.

– работать с сгенерированными по специальным правилам открытым и закрытыми ключами.

– реализовать расширенный алгоритм Евклида.

# 2 ХОД РАБОТЫ

Алгоритм был реализован на языке PHP. Для выполнения задачи был создан объект, содержащий все методы, необходимые для шифрования и дешифрования данных. Алгоритм автоматически генерирует открытый и закрытый ключ. Далее алгоритм применяет алгоритм шифрования для входных данных и возвращает его. Далее скрипт выполняет дешифровку данных используя закрытый ключ.

Результат полного выполнения скрипта можно увидеть на рисунке 1.

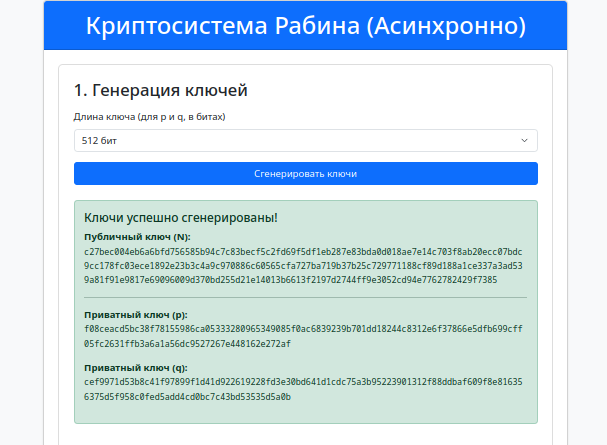


Рисунок 1 – Вывод генерации ключей

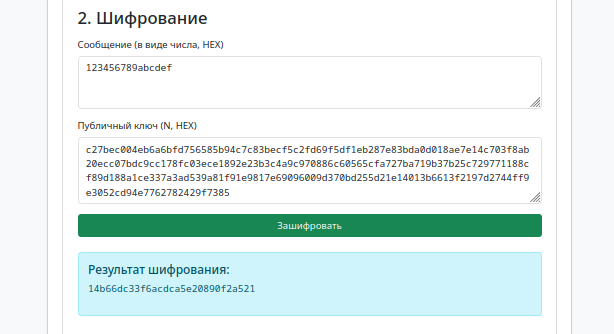


Рисунок 2 – Вывод шифрования

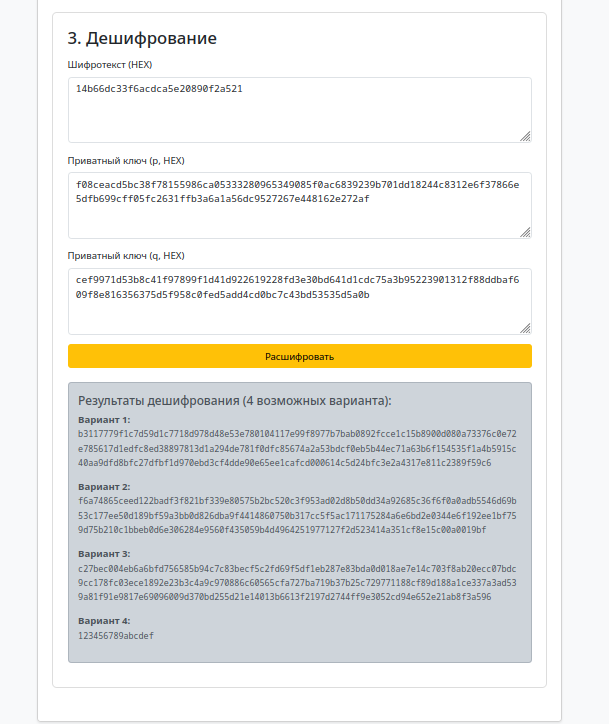


Рисунок 3 – Вывод шифрования

Как видно на рисунках 1, 2 и 3, скрипт успешно справился с шифровкой и дешифровкой информации. Одно из полей вывода содержит оригинальное значение.

# ЗАКЛЮЧЕНИЕ

В ходе выполнения лабораторной работы были изучены теоретические основы асимметричных шифров на примере криптосистемы Рабина. Также на PHP был реализована его реализация с возможностью шифрование и расшифровки данных, используя набор открытого и закрытых ключей. Таким образом, поставленные цели были достигнуты: алгоритм был реализован, проведены эксперименты с его использованием, а также закреплены навыки программной реализации асимметричных блочных шифров.

# ПРИЛОЖЕНИЕ А

# (ОБЯЗАТЕЛЬНОЕ)

# ИСХОДНЫЙ КОД ПРОГРАММНОГО ПРОДУКТА

<?php

namespace App;

use Exception;

use GMP;

class RabinSystem

{

/\*\*

\* @param GMP $a

\* @param GMP $b

\* @return GMP[]

\*/

private static function extendedGcd(GMP $a, GMP $b): array

{

// [g, s, t], g = НОД, as + bt = g

return gmp\_gcdext($a, $b);

}

/\*\*

\* r ≡ mp (mod p) и r ≡ mq (mod q).

\*

\* @param GMP $mp

\* @param GMP $p

\* @param GMP $mq

\* @param GMP $q

\* @return GMP

\*/

private static function crt(GMP $mp, GMP $p, GMP $mq, GMP $q): GMP

{

list($gcd, $yp, $yq) = self::extendedGcd($p, $q);

$n = gmp\_mul($p, $q);

// Формула: (mp\*yq\*q + mq\*yp\*p) mod n

$term1 = gmp\_mul(gmp\_mul($mp, $yq), $q);

$term2 = gmp\_mul(gmp\_mul($mq, $yp), $p);

return gmp\_mod(gmp\_add($term1, $term2), $n);

}

/\*\*

\*

\* @param GMP $mp

\* @param GMP $p

\* @param GMP $mq

\* @param GMP $q

\* @return GMP

\*/

private static function mCrt(GMP $mp, GMP $p, GMP $mq, GMP $q): GMP

{

list($gcd, $yp, $yq) = self::extendedGcd($p, $q);

$n = gmp\_mul($p, $q);

// Формула: (mp\*yq\*q - mq\*yp\*p) mod n

$term1 = gmp\_mul(gmp\_mul($mp, $yq), $q);

$term2 = gmp\_mul(gmp\_mul($mq, $yp), $p);

return gmp\_mod(gmp\_sub($term1, $term2), $n);

}

/\*\*

\*

\* @param int $bitLength Длина ключа в битах для каждого простого числа.

\* @return array ['public' => GMP, 'private' => ['p' => GMP, 'q' => GMP]]

\*/

public static function generateKeys(int $bitLength = 512): array

{

$four = gmp\_init(4);

$three = gmp\_init(3);

// Генерируем простое число p, такое что p ≡ 3 (mod 4)

do {

$p = gmp\_nextprime(gmp\_random\_bits($bitLength));

} while (gmp\_cmp(gmp\_mod($p, $four), $three) !== 0);

// Генерируем простое число q, такое что q ≡ 3 (mod 4)

do {

$q = gmp\_nextprime(gmp\_random\_bits($bitLength));

} while (gmp\_cmp(gmp\_mod($q, $four), $three) !== 0);

$n = gmp\_mul($p, $q); // Публичный ключ

return [

'public' => $n,

'private' => ['p' => $p, 'q' => $q]

];

}

/\*\*

\*

\* @param GMP $message Сообщение (представленное как число)

\* @param GMP $publicKey Публичный ключ (N)

\* @return GMP Шифротекст

\* @throws Exception

\*/

public static function encrypt(GMP $message, GMP $publicKey): GMP

{

if (gmp\_cmp($message, $publicKey) >= 0) {

throw new Exception("Сообщение должно быть меньше публичного ключа N.");

}

// c = m^2 mod N

return gmp\_powm($message, gmp\_init(2), $publicKey);

}

/\*\*

\*

\* @param GMP $ciphertext Шифротекст

\* @param array $privateKey ['p' => GMP, 'q' => GMP]

\* @return GMP[] Массив из четырех возможных исходных сообщений.

\*/

public static function decrypt(GMP $ciphertext, array $privateKey): array

{

$p = $privateKey['p'];

$q = $privateKey['q'];

$n = gmp\_mul($p, $q);

$exp\_p = gmp\_div\_q(gmp\_add($p, gmp\_init(1)), gmp\_init(4));

$mp = gmp\_powm($ciphertext, $exp\_p, $p);

$exp\_q = gmp\_div\_q(gmp\_add($q, gmp\_init(1)), gmp\_init(4));

$mq = gmp\_powm($ciphertext, $exp\_q, $q);

$r1 = self::crt($mp, $p, $mq, $q);

$r2 = gmp\_sub($n, $r1);

$s1 = self::mCrt($mp, $p, $mq, $q);

$s2 = gmp\_sub($n, $s1);

return [$r1, $r2, $s1, $s2];

}

}