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 №1

на тему

**Симметричная криптография.** **Стандарт шифрования ГОСТ 28147-89.** **Режим** **гаммирования с обратной связью**

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 5](#_Toc207722282)

[Приложение А (обязательное) исходный код программного продукта 6](#_Toc207722283)

# 1 ЦЕЛЬ РАБОТЫ

Целью данной лабораторной работы является изучение принципов симметричного блочного шифрования на примере стандарта шифрования

ГОСТ 28147-89 и его реализации в режиме простой замены.

В рамках работы требуется разработать программные средство для шифрования и дешифрования текстовых файлов, закрепить навыки работы с

блочными криптографическими алгоритмами и освоить процесс преобразования данных с использованием фиксированных ключей и таблиц подстановки (S-блоков).

Результатом выполнения работы должен быть скрипт, который позволяет:

– зашифровать текстовый файл при помощи ГОСТ 28147-89 в режиме промтой замены.

– расшифровать зашифрованный файл обратно в исходный вид.

– работать с произвольным ключом (256 бит) и встроенными S-блоками.

– автоматически дополнять данные до кратности 64 бит.

# 2 ХОД РАБОТЫ

Алгоритм был реализован на языке PHP. Для выполнения задачи был создан объект, содержащий все методы, необходимые для шифрования и дешифрования файлов. Алгоритм автоматически генерирует случайные ключ и вектор инициализации. Далее алгоритм считывает данные из входного файла и применяет алгоритм шифрования. Полученный результат он записывает в бинарный файл с приставкой enc. Далее скрипт считывает данные с файла и запускает процесс расшифровки.

Результат полного выполнения скрипта можно увидеть на рисунке 1.

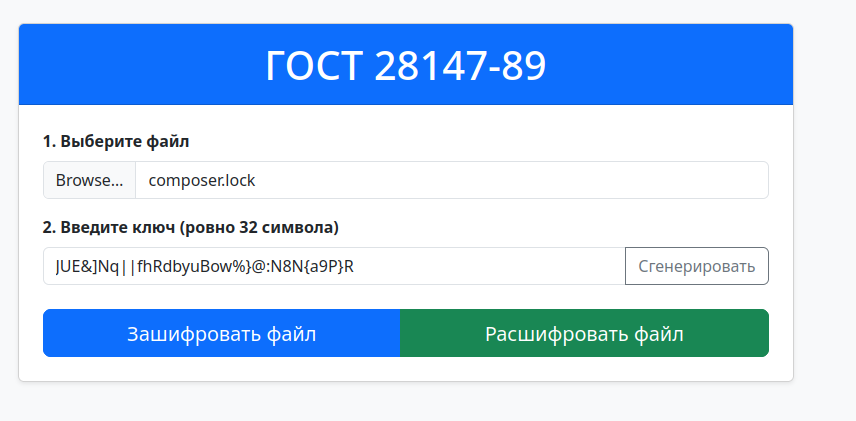


Рисунок 1 – Вывод работы программы

Как видно на рисунке 1, скрипт успешно справился с шифровкой и дешифровкой информации.

# ЗАКЛЮЧЕНИЕ

В ходе выполнения лабораторной работы были изучены теоретические основы симметричных блочных шифров на примере алгоритма ГОСТ 28147-89, построенного по схеме Фейстеля. Также на PHP был реализован данные гост в режиме простой замены. Таким образом, поставленные цели были достигнуты: алгоритм был реализован, проведены эксперименты с его использованием, а также закреплены навыки программной реализации симметричных блочных шифров.

# ПРИЛОЖЕНИЕ А

# (ОБЯЗАТЕЛЬНОЕ)

# ИСХОДНЫЙ КОД ПРОГРАММНОГО ПРОДУКТА

<?php

namespace App;

class GostEncrypt

{

private const DEFAULT\_S\_BOXES = [

[4, 10, 9, 2, 13, 8, 0, 14, 6, 11, 1, 12, 7, 15, 5, 3],

[14, 11, 4, 12, 6, 13, 15, 10, 2, 3, 8, 1, 0, 7, 5, 9],

[5, 8, 1, 13, 10, 3, 4, 2, 14, 15, 12, 7, 6, 0, 9, 11],

[7, 13, 10, 1, 0, 8, 9, 15, 14, 4, 6, 12, 11, 2, 5, 3],

[6, 12, 7, 1, 5, 15, 13, 8, 4, 10, 9, 14, 0, 3, 11, 2],

[4, 11, 10, 0, 7, 2, 1, 13, 3, 6, 8, 5, 9, 12, 15, 14],

[13, 11, 4, 1, 3, 15, 5, 9, 0, 10, 14, 7, 6, 8, 2, 12],

[1, 15, 13, 0, 5, 7, 10, 4, 9, 2, 3, 14, 6, 11, 8, 12]

];

/\*\*

\* @param string $key

\* @return int[]

\*/

private function prepareSubkeys(string $key): array

{

$subkeys = [];

for ($i = 0; $i < 32; $i += 4) {

$subkeys[] = unpack('N', substr($key, $i, 4))[1];

}

$roundSubkeys = [];

for ($round = 0; $round < 32; $round++) {

if ($round < 24) {

$roundSubkeys[] = $subkeys[$round % 8];

} else {

$roundSubkeys[] = $subkeys[7 - ($round % 8)];

}

}

return $roundSubkeys;

}

/\*\*

\* @param int|null $data

\* @param int $subkey

\* @param array $sBoxes

\*

\* @return int

\*/

private function feistelFunction(?int $data, int $subkey, array $sBoxes): int

{

$temp = ($data + $subkey) & 0xFFFFFFFF;

$result = 0;

for ($i = 0; $i < 8; $i++) {

$sBoxIndex = ($temp >> (4 \* $i)) & 0xF;

$sBoxValue = $sBoxes[$i][$sBoxIndex];

$result |= ($sBoxValue << (4 \* $i));

}

return (($result << 11) | ($result >> 21)) & 0xFFFFFFFF;

}

/\*\*

\* @param string $block

\* @param string $key

\* @return string

\*/

private function gostEncryptBlock(string $block, string $key): string

{

$sBoxes = self::DEFAULT\_S\_BOXES;

$subkeys = $this->prepareSubkeys($key);

$parts = unpack('N2', $block);

$left = $parts[1];

$right = $parts[2];

for ($round = 0; $round < 32; $round++) {

$subkey = $subkeys[$round];

$fResult = $this->feistelFunction($right, $subkey, $sBoxes);

$newRight = $left ^ $fResult;

$left = $right;

$right = $newRight;

}

[$left, $right] = [$right, $left];

return pack('NN', $left, $right);

}

/\*\*

\* @param string $block

\* @param string $key

\* @return string

\*/

private function gostDecryptBlock(string $block, string $key): string

{

$sBoxes = self::DEFAULT\_S\_BOXES;

$subkeys = $this->prepareSubkeys($key);

$parts = unpack('N2', $block);

$left = $parts[1];

$right = $parts[2];

for ($round = 31; $round >= 0; $round--) {

$subkey = $subkeys[$round];

$fResult = $this->feistelFunction($right, $subkey, $sBoxes);

$newRight = $left ^ $fResult;

$left = $right;

$right = $newRight;

}

[$left, $right] = [$right, $left];

return pack('NN', $left, $right);

}

/\*\*

\* @param string $data

\* @param string $key

\* @return string

\*/

public function encrypt(string $data, string $key): string

{

$paddingLength = 8 - (strlen($data) % 8);

if ($paddingLength === 0) {

$paddingLength = 8;

}

$paddedData = $data . str\_repeat(chr($paddingLength), $paddingLength);

$encryptedBlocks = [];

for ($i = 0; $i < strlen($paddedData); $i += 8) {

$block = substr($paddedData, $i, 8);

$encryptedBlocks[] = $this->gostEncryptBlock($block, $key);

}

return implode('', $encryptedBlocks);

}

/\*\*

\* @param string $encryptedData

\* @param string $key

\* @return string|false

\*/

public function decrypt(string $encryptedData, string $key)

{

$decryptedBlocks = [];

for ($i = 0; $i < strlen($encryptedData); $i += 8) {

$block = substr($encryptedData, $i, 8);

$decryptedBlocks[] = $this->gostDecryptBlock($block, $key);

}

$decryptedData = implode('', $decryptedBlocks);

$lastByte = substr($decryptedData, -1);

if ($lastByte === false) {

return false;

}

$paddingLength = ord($lastByte);

if ($paddingLength > 0 && $paddingLength <= 8) {

$padding = substr($decryptedData, -$paddingLength);

if ($padding === str\_repeat(chr($paddingLength), $paddingLength)) {

return substr($decryptedData, 0, -$paddingLength);

}

}

return $decryptedData;

}

}