Министерство образования Республики Беларусь

Учреждение образования «Белорусский государственный университет   
информатики и радиоэлектроники»

Факультет компьютерных систем и сетей

Кафедра информатики

Дисциплина «Методы защиты информации»

**ОТЧЕТ**

к лабораторной работе

на тему:

**«Асимметричная криптография. Криптосистема Рабина»**

|  |  |
| --- | --- |
|  | Выполнил:  Слуцкий Никита Сергеевич,  студент группы 053505 |
|  | Проверил: Лещенко Евгений Александрович, ассистент каф. Информатики |

Минск 2023

СОДЕРЖАНИЕ

[Введение 3](#_Toc146449686)

[1 Краткие теоретические сведения 4](#_Toc146449687)

[1.1 Область применения 4](#_Toc146449688)

[1.2 Алгоритм шифрования блока 4](#_Toc146449689)

[1.3 Гаммирование с обратной связью 5](#_Toc146449690)

[2 Ход выполнения работы 6](#_Toc146449691)

[Заключение 7](#_Toc146449692)

[Приложение А Листинг кода 8](#_Toc146449693)

# ВВЕДЕНИЕ

Целью данной лабораторной работы является реализовать программное средство шифрования и дешифрования текстовых файлов при помощи алгоритма криптосистемы Рабина на выбранном языке программирования.

# 1 Краткие теоретические сведения

## 1.1 Область применения

Настоящий стандарт определяет семейство криптографических алгоритмов шифрования и контроля целостности, которые используются для защиты информации при ее хранении, передаче и обработке.

Настоящий стандарт применяется при разработке средств криптографической защиты информации.

## Алгоритм шифрования блока

Входными данными алгоритмов зашифрования и расшифрования являются блок 𝑋 длиной 16 байт и ключ 𝜃 длиной 32 байта.

Входные данные подготавливаются следующим образом.

Слово 𝑋 записывается в виде 𝑋 = 𝑋1 || 𝑋2 || 𝑋3 || 𝑋4, где 𝑋𝑖 имеет длину 4 байта.

Ключ 𝜃 записывается в виде 𝜃 = 𝜃1 || 𝜃2 || … || 𝜃8, 𝜃i с длиной 4 байта. И определяются тактовые ключи 𝐾1 = 𝜃1, 𝐾2 = 𝜃2, . . . , 𝐾8 = 𝜃8, 𝐾9 = 𝜃1, 𝐾10 = 𝜃2, . . . , 𝐾56 = 𝜃8.

Используются переменные 𝑎, 𝑏, 𝑐, 𝑑, 𝑒. Переменные a, b,   
c, d соответствуют частям исходного блока X, описанного выше.

Для зашифрования блока 𝑋 на ключе 𝜃 выполняются следующие шаги:

* установить 𝑎 ← 𝑋1, 𝑏 ← 𝑋2, 𝑐 ← 𝑋3, 𝑑 ← 𝑋4;
* для 𝑖 = 1, 2, . . . , 8 выполнить следующие шаги, проиллюстрированные на рисунке 1;
* 𝑏 ← 𝑏 ⊕ 𝐺5(𝑎 ⊕ 𝐾7𝑖−6);
* 𝑐 ← 𝑐 ⊕ 𝐺21(𝑑 ⊕ 𝐾7𝑖−5);
* 𝑎 ← 𝑎 - 𝐺13(𝑏 ⊕ 𝐾7𝑖−4);
* 𝑒 ← 𝐺21(𝑏 ⊕ 𝑐 ⊕ 𝐾7𝑖−3) ⊕ ⟨𝑖⟩32;
* 𝑏 ← 𝑏 ⊕ 𝑒;
* 𝑐 ← 𝑐 - 𝑒;
* 𝑑 ← 𝑑 ⊕ 𝐺13(𝑐 ⊕ 𝐾7𝑖−2);
* 𝑏 ← 𝑏 ⊕ 𝐺21(𝑎 ⊕ 𝐾7𝑖−1);
* 𝑐 ← 𝑐 ⊕ 𝐺5(𝑑 ⊕ 𝐾7𝑖);
* 𝑎 ↔ 𝑏;
* 𝑐 ↔ 𝑑;
* 𝑏 ↔ 𝑐.

После проделывания операций необходимо снова склеить в блок преобразованные значения a, b, c, d и вернуть зашифрованный блок.

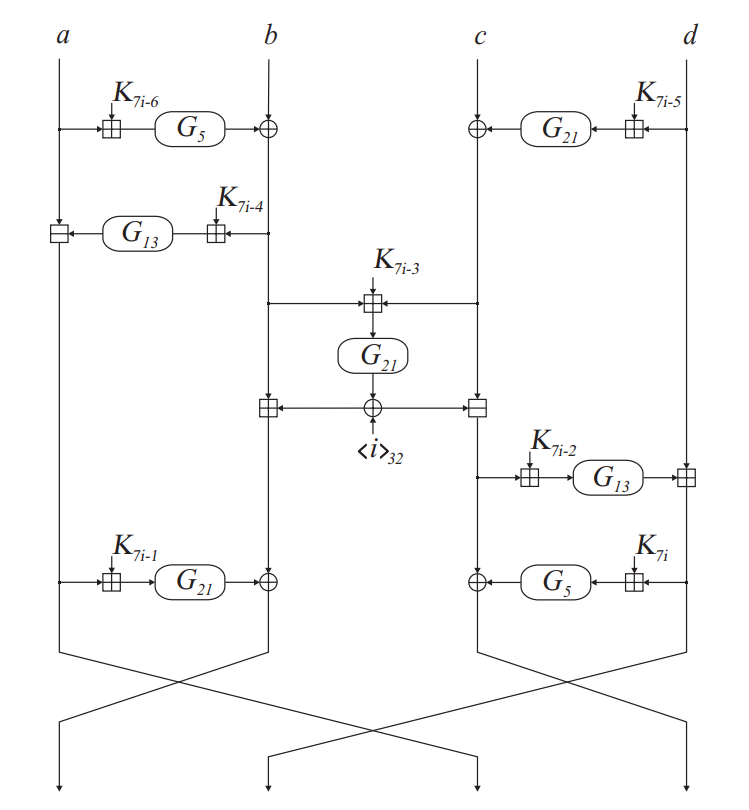


Рисунок 1 – Вычисления на i-м такте шифрования

Преобразование 𝐺𝑟 : {0, 1}32 → {0, 1}32 ставит в соответствие слову   
𝑢 = 𝑢1 || 𝑢2 || 𝑢3 || 𝑢4, 𝑢𝑖 ∈ {0, 1}8 , слово 𝐺𝑟(𝑢) = RotHi𝑟 (𝐻(𝑢1) || 𝐻(𝑢2) ‖ 𝐻(𝑢3) || 𝐻(𝑢4)). RotHi(𝑢) для 𝑢 ∈ {0, 1}8𝑛 – это слово ShHi(𝑢) ⊕ ShLo8𝑛−1(𝑢), где:

* ShHi(𝑢) = ⟨2¯𝑢⟩8𝑛;
* ShLo(𝑢) = ⟨⌊𝑢¯ 2⌋⟩8n;
* ⌊𝑧⌋ – для вещественного 𝑧 максимальное целое, не превосходящее z;
* ⟨𝑈⟩8n – слово 𝑢 ∈ {0, 1}8𝑛 такое, что 𝑢¯ = 𝑈 mod 28n.

## 1.3 Гаммирование с обратной связью

В режиме гаммирования с обратной связью текст всё так же разбивается на блоки. Вводится дополнительный параметр – начальный гамма-блок. Шифруется с помощью метода простой замены сам гамма-блок, а затем поэлементно складывается по модулю 2 с блоком. Полученный результат – и есть результат шифрования блока. Гамма-блок, в свою очередь, также перезаписывается получившимся значением.

Таким образом на каждой итерации значение гаммы разное, оно получается из предыдущего шага. Отсюда и название режима. На рисунках 2, 2 изображена схема шифрования и дешифрования с использованием этого режима.

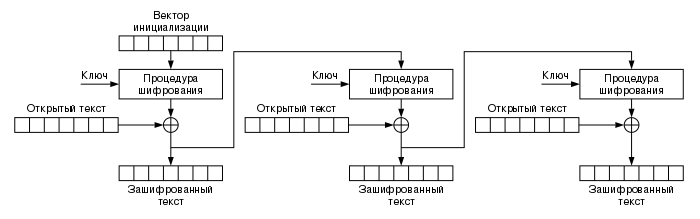


Рисунок 2 – Схема шифрования в режиме гаммирования с обратной связью

# 2 Ход выполнения работы

Программное средство написано на языке программирования C++. В файле STB\_34\_101\_31\_2011/core/core.cpp находятся, соответственно, тела функций, отвечающих за базовое шифрование блока и вспомогательные действия. В других двух папках в STB\_34\_101\_31\_2011 находятся функции для шифрования в режимах простой замены и гаммирования с обратной связью. В такой же группе файлов utils находятся вспомогательные функции для ввода, вывода, работы со строками и массивами. Программа написана на современном стандарте языка C++ 17 с использованием контейнеров STL, итераторов и современного «синтаксического сахара», а также некоторых практик по именованию из Google Style Guide для написания читаемого кода.

В приложении с листингом кода предоставлена вырезка кода с функцией шифрования в режиме гаммирования с обратной связью.

Реализованное программное средство имеет возможность считать файл, зашифровать или расшифровать его содержимое в режиме простой замены или гаммирования с обратной связью и сохранить преобразованный результат. Таким образом после прогона средство шифрует либо дешифрует файл.

# Заключение

В ходе выполнения данной работы было произведено ознакомление с теорией об алгоритме шифрования СТБ 34.101.31-2011, реализовано программное средство для шифрования и расшифрования файлов с использованием данного шифрования в режиме гаммирования с обратной связью.

Цель лабораторной работы, поставленная во введении настоящего отчёта, можно считать достигнутой.

# ПРИЛОЖЕНИЕ А

(Листинг кода)

std::vector<uint8\_t> encrypt\_by\_stb\_34\_101\_31\_2011\_with\_reversal\_gamma(

const std::vector<uint8\_t> &text,

const std::vector<uint8\_t> &key,

const std::vector<uint8\_t> &initial\_gamma

)

{

const std::vector<uint32\_t> subkeys{get\_subkeys(key)};

const std::vector<std::vector<uint8\_t>> blocks{split\_vector\_on\_blocks(text, kEncryptedBlockLength)};

std::vector<uint8\_t> response{};

std::vector<uint8\_t> gamma(std::cbegin(initial\_gamma), std::cend(initial\_gamma));

for (const auto &block : blocks)

{

const std::vector<uint8\_t> crypted\_gamma{encrypt\_block(gamma, subkeys)};

const std::vector<uint8\_t> block\_xored\_with\_gamma{apply\_gamma(block, crypted\_gamma)};

gamma = slice\_vector(block\_xored\_with\_gamma, 0, std::size(block\_xored\_with\_gamma));

for (const auto &byte : block\_xored\_with\_gamma)

{

response.push\_back(byte);

}

}

return response;

}