Министерство образования Республики Беларусь

Учреждение образования

БЕЛОРУССКИЙ ГОССУДАРСТВЕННЫЙ УНИВЕРСИТЕТ

ИНФОРМАТИКИ И РАДИОЭЛЕКТРОННИКИ

КАФЕДРА ИНФОРМАТИКИ

Отчёт по лабораторной работе №5

По дисциплине «Методы защиты информации»

По теме «Хэш-функции»

Выполнил:

студент гр. 653502

Максименко П.В.

Проверил:

Артемьев В. С.

Минск 2019

1. Постановка зада4и

Написать программу, реализующую алгоритм HMAC.

2. Теорети4еская 4асть

HMAC ([сокращение](https://ru.wikipedia.org/wiki/%D0%90%D0%B1%D0%B1%D1%80%D0%B5%D0%B2%D0%B8%D0%B0%D1%82%D1%83%D1%80%D0%B0) от [англ.](https://ru.wikipedia.org/wiki/%D0%90%D0%BD%D0%B3%D0%BB%D0%B8%D0%B9%D1%81%D0%BA%D0%B8%D0%B9_%D1%8F%D0%B7%D1%8B%D0%BA) *hash-based message authentication code*, код [аутентификации](https://ru.wikipedia.org/wiki/%D0%90%D1%83%D1%82%D0%B5%D0%BD%D1%82%D0%B8%D1%84%D0%B8%D0%BA%D0%B0%D1%86%D0%B8%D1%8F) (проверки подлинности) сообщений, использующий [хеш](https://ru.wikipedia.org/wiki/%D0%A5%D0%B5%D1%88-%D1%81%D1%83%D0%BC%D0%BC%D0%B0)-[функции](https://ru.wikipedia.org/wiki/%D0%A4%D1%83%D0%BD%D0%BA%D1%86%D0%B8%D1%8F_(%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5))) — в [информатике](https://ru.wikipedia.org/wiki/%D0%98%D0%BD%D1%84%D0%BE%D1%80%D0%BC%D0%B0%D1%82%D0%B8%D0%BA%D0%B0) ([криптографии](https://ru.wikipedia.org/wiki/%D0%9A%D1%80%D0%B8%D0%BF%D1%82%D0%BE%D0%B3%D1%80%D0%B0%D1%84%D0%B8%D1%8F)), один из механизмов проверки целостности информации, позволяющий гарантировать то, что данные, передаваемые или хранящиеся в ненадёжной среде, не были изменены посторонними лицами (см. [человек посередине](https://ru.wikipedia.org/wiki/%D0%A7%D0%B5%D0%BB%D0%BE%D0%B2%D0%B5%D0%BA_%D0%BF%D0%BE%D1%81%D0%B5%D1%80%D0%B5%D0%B4%D0%B8%D0%BD%D0%B5)). Механизм HMAC использует [MAC](https://ru.wikipedia.org/wiki/%D0%98%D0%BC%D0%B8%D1%82%D0%BE%D0%B2%D1%81%D1%82%D0%B0%D0%B2%D0%BA%D0%B0), описан в [RFC 2104](https://tools.ietf.org/html/rfc2104), в стандартах организаций [ANSI](https://ru.wikipedia.org/wiki/ANSI), [IETF](https://ru.wikipedia.org/wiki/IETF), [ISO](https://ru.wikipedia.org/wiki/ISO) и [NIST](https://ru.wikipedia.org/wiki/NIST). MAC — стандарт, описывающий способ обмена данными и способ проверки целостности передаваемых данных с использованием секретного ключа. Два клиента, использующие HMAC, как правило, разделяют общий секретный ключ. HMAC — надстройка над MAC; механизм обмена данными с использованием секретного ключа (как в MAC) и [хеш-функций](https://ru.wikipedia.org/wiki/%D0%A5%D0%B5%D1%88-%D1%84%D1%83%D0%BD%D0%BA%D1%86%D0%B8%D1%8F). В зависимости от используемой хеш-функции выделяют HMAC-[MD5](https://ru.wikipedia.org/wiki/MD5), HMAC-[SHA1](https://ru.wikipedia.org/wiki/SHA-1), HMAC-[RIPEMD128](https://ru.wikipedia.org/wiki/RIPEMD-128), HMAC-[RIPEMD160](https://ru.wikipedia.org/wiki/RIPEMD-160) и т. п.

3. Блок-схема

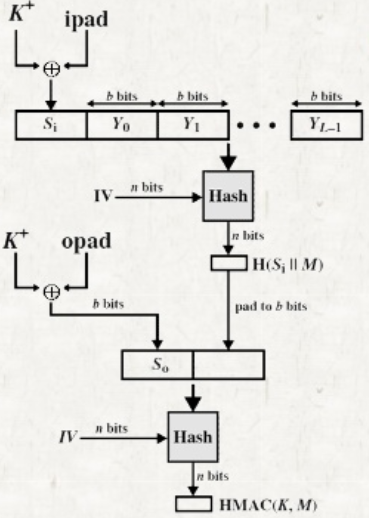


Рис 3.1. Схема работы алгоритма.

4. Ход программы



Рис 4.1. Данные, поступающие на вход

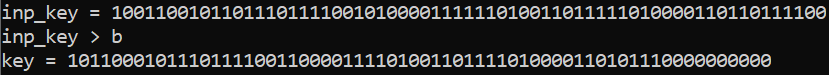


Рис 4.2. Трансформация клю4а



Рис 4.3. Создание si и s0 блоков для шифрации



Рис 4.4. Зашифрованные данные

5. Код программы

|  |
| --- |
| string key\_xor(string str, string key, int mod) |
|  | { |
|  | string final\_str = str; |
|  |  |
|  | for (int i = 0; i < key.length(); i++) { |
|  | if (final\_str[i + mod] == '0' && key[i] == '0') { |
|  | final\_str[i + mod] = '0'; |
|  | } |
|  | else if (final\_str[i + mod] == '0' && key[i] == '1') { |
|  | final\_str[i + mod] = '1'; |
|  | } |
|  | else if (final\_str[i + mod] == '1' && key[i] == '0') { |
|  | final\_str[i + mod] = '1'; |
|  | } |
|  | else if (final\_str[i + mod] == '1' && key[i] == '1') { |
|  | final\_str[i + mod] = '0'; |
|  | } |
|  | } |
|  |  |
|  | return final\_str; |
|  | } |
|  |  |
|  |  |
|  | std::string HMAC(string inp\_data, string inp\_key) |
|  | { |
|  | std::cout << "inp\_key = " << inp\_key << std::endl; |
|  |  |
|  | // check inp\_key length and correct it |
|  | int b = 64; |
|  | string key; |
|  | if (inp\_key.length() == b) |
|  | { |
|  | key = inp\_key; |
|  | std::cout << "inp\_key = b" << std::endl; |
|  | } |
|  | else if (inp\_key.length() < b) |
|  | { |
|  | key = inp\_key; |
|  | while (key.length() < b) |
|  | { |
|  | key = key + "0"; |
|  | } |
|  | std::cout << "inp\_key < b" << std::endl; |
|  | } |
|  | else if (inp\_key.length() > b) |
|  | { |
|  | key = inp\_key; |
|  | while (key.length() > b) |
|  | { |
|  | // hash-function |
|  | for (int i = 0; i < key.length(); i = i + 2) { |
|  | if (int(inp\_key[i]) % 2 == 0) |
|  | { |
|  | key.erase(i, 1); |
|  | } |
|  | } |
|  | } |
|  | while (key.length() < b) |
|  | { |
|  | key = key + "0"; |
|  | } |
|  | std::cout << "inp\_key > b" << std::endl; |
|  | } |
|  | std::cout << "key = " << key << std::endl; |
|  |  |
|  | string si = "000000000000000000000000000000000000000000000000000000000000000"; |
|  | string ipad = "111110010111100011110010101101110010100110111110111001111100111"; |
|  | for (int i = 0; i < b; i++) { |
|  | si = key\_xor(key, ipad, 0); |
|  | } |
|  | std::cout << "si = " << si << std::endl; |
|  |  |
|  | string s0 = "000000000000000000000000000000000000000000000000000000000000000"; |
|  | string opad = "000010010111100011110010101101010010100110100010111001100000111"; |
|  | for (int i = 0; i < b; i++) { |
|  | s0 = key\_xor(key, opad, 0); |
|  | } |
|  | std::cout << "s0 = " << s0 << std::endl; |
|  |  |
|  | string data = inp\_data + si; |
|  | // hash-function |
|  | for (int i = 0; i < data.length(); i = i + 2) { |
|  | if (int(data[i]) % 2 == 0) |
|  | { |
|  | data.erase(i, 1); |
|  | } |
|  | } |
|  | data = data + s0; |
|  | // hash-function |
|  | for (int i = 0; i < data.length(); i = i + 2) { |
|  | if (int(data[i]) % 2 == 0) |
|  | { |
|  | data.erase(i, 1); |
|  | } |
|  | } |
|  |  |
|  | std::cout << "data atfer conversion = " << data << std::endl; |
|  |  |
|  | return data; |
|  | } |
|  |  |
|  |  |
|  | int main() |
|  | { |
|  | SetConsoleCP(1251); |
|  | SetConsoleOutputCP(1251); |
|  |  |
|  | // lab5 |
|  | // HMAC |
|  | string key = "10011001011011101111001010000111111010011011111010000110110111100"; |
|  | string data = "00001110001011010001110100001110001011100001111000001110110001111"; |
|  | HMAC(data, key); |
|  |  |
|  | return 0; |
|  | } |

6. Вывод

В ходе выполнения лабораторной работы был успешно реализован алгоритм HMAC. Стоит также отметить, 4то безопасность любой функции MAC на основе встроенных хеш-функций зависит от криптостойкости базовой хеш-функции.

Также, были выделены следующие преимущества HMAC:

* возможность использования хеш-функций, уже имеющихся в программном продукте;
* отсутствие необходимости внесения изменений в реализации, существующих хеш-функции (внесение изменений может привести к ухудшению производительности и криптостойкости);
* возможность замены хеш-функции в случае появления более безопасной или более быстрой хеш-функции.