Міністерство освіти та науки України

Харківський національний універсітет радіоелектроніки

Факультет КОМП’ЮТЕРНОЇ ІНЖЕНЕРІЇ ТА УПРАВЛІННЯ

Кафедра БЕЗПЕКИ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ

**КУРСОВИЙ ПРОЕКТ**

**ПОЯСНЮВАЛЬНА ЗАПИСКА**

**З дисципліни “Програмування мовою Assembler ”**

**ТЕМА: “Хеш-функція UMAC”**

Виконав: Перевірив:

Ст. гр. \*\*\*\* \*\*\*\*\*\*\*\*\*\*\*\*\*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Харків – 2008

Харківський національний університет радіоелектроніки

Факультет: КІУ Кафедра: Безпеки інформаційних технологій

Спеціальність: “Захист інформації з обмеженим доступом та автоматизація її обробки”

Дисципліна: Програмування мовою “Assembler”

ЗАТВЕРДЖУЮ

Зав. кафедри БІТ проф. \*\*\*\*\*\*\*

“\_\_\_\_\_“ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ 2008р.

**ЗАВДАННЯ**

**НА КУРСОВИЙ ПРОЕКТ (РОБОТУ)**

Студентові \_\_\_\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

(прізвище, ім'я, по батькові)

1. Тема проекту (роботи) \_\_\_\_Хеш-функція UMAC\_\_\_\_\_\_\_\_\_\_\_\_
2. Термін здачі студентом закінченого проекту (роботи)\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_19.01.2008р.\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_
3. Вихідні дані до проекту: \_\_\_\_\_\_\_\_Дані з предметної галузі, методичні вказівки.\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_
4. Зміст пояснювальної записки (перелік питань, що їх потрібно розробити) \_\_\_\_Вступ, Аналіз предметної галузі, Опис програми, Основні особливості MASM32, інструкція користувача, Висновок.\_\_\_\_
5. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень, плакатів)

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

1. Основна література та джерела

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

7. Дата видачі завдання \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

8. Дата здачі завдання \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Керівник проекту (роботи)\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(підпис) (посада, прізвище, ім'я, по батькові)

Завдання прийняв до виконання \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(підпис студента)

Студент\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(підпис)

**СОДЕРЖАНИЕ**

ВВЕДЕНИЕ ………………………………………………………………….5

1Анализ предметной области……………………………………………….7

1.1Основные особенности среды MASM32 ……………………………….7

1.2Анализ математических методов ……………………………………….8

2 Постановка задачи ………………………………………………………...9

3 Описание программы ……………………………………………………..10

3.1 Общие сведения

3.2 Назначение и логическая структура

4 Инструкции пользователя

Выводы

Перечень ссылок

Приложение А Экранные формы программы

Приложение Б Тексты модулей программы

**ВВЕДЕНИЕ**

Продолжающееся развитие компьютерных технологий и повсеместное внедрение бизнес-процессов с использованием глобальной сети Интернет коренным образом изменяет устоявшиеся способы ведения бизнеса. Системы корпоративной безопасности, обеспечивающие бизнес, тоже не могут оставаться в стороне.

В настоящее время, например, средства электронной почты используются не только для общения между людьми, а и для передачи контрактов и конфиденциальной финансовой информации. Web-серверы используются не только для рекламных целей, но и для распространения программного обеспечения и электронной коммерции. Электронная почта, доступ к Web-серверу, электронная коммерция, VPN требуют применения дополнительных средств для обеспечения конфиденциальности, аутентификации, контроля доступа, целостности и идентификации. В качестве таких средств сегодня повсеместно используются средства криптографической защиты и инфраструктура открытых ключей (Public Key Infrastructure, PKI).

Система криптографической защиты должна обеспечивать:

1. Конфиденциальность. Информация должна быть защищена от не-санкционированного прочтения как при хранении, так и при передаче. Если сравнивать с бумажной технологией, то это аналогично запечатыванию информации в конверт. Содержание становится известно только после того, как будет открыт конверт. В системах криптографической защиты обеспечивается шифрованием.

2. Контроль доступа. Информация должна быть доступна только для того, для кого она предназначена. Если сравнивать с бумажной технологией, то только разрешенный получатель может открыть запечатанный конверт. В системах криптографической защиты обеспечивается шифрованием.

3. Аутентификацию. Возможность однозначно идентифицировать отправителя. Если сравнивать с бумажной технологией, то это аналогично подписи отправителя. В системах крипто-графической защиты обеспечивается электронной цифровой подписью и сертификатом.

4. Целостность. Информация должна быть защищена от несанкционированной модификации как при хранении, так и при передаче. В системах криптографической защиты обеспечивается электронной цифровой подписью.

5. Неопровергаемость. Отправитель не может отказаться от совершенного действия. Если сравнивать с бумажной технологией, то это аналогично предъявлению отправителем па-спорта перед выполнением действия. В системах криптографической защиты обеспечивается электронной цифровой подписью и сертификатом.

Криптографические хэш-функции играют фундаментальную роль в современной криптографии. Говоря в общем хэш-функция h отображает двоичные строки произвольной конечной длины в выходы небольшой (например, 64, 128, 160,192, 224, 256, 384, 512) фиксированной длины называемые хэш-величинами за полиномиальное время.

Область применения хэш-функции четко неоговорена: используется “для реализации процедур электронной цифровой подписи, при передаче, обработке и хранении информации в автоматизированных системах”.

Среда программирования MASM32 позволяет создавать тексты программ, компилировать их, находить ошибки и оперативно их исправлять; компоновать программы из отдельных частей, включая стандартные модули, отлаживать и выполнять отлаженную программу.

Используя перечисленные возможности, можно создавать различные прикладные программы, например, такие, как программа, написанная при выполнении данной курсовой работы.

**1. АНАЛИЗ ПРЕДМЕТНОЙ ОБЛАСТИ**

**1.1 MAC-код аутентификации сообщения**

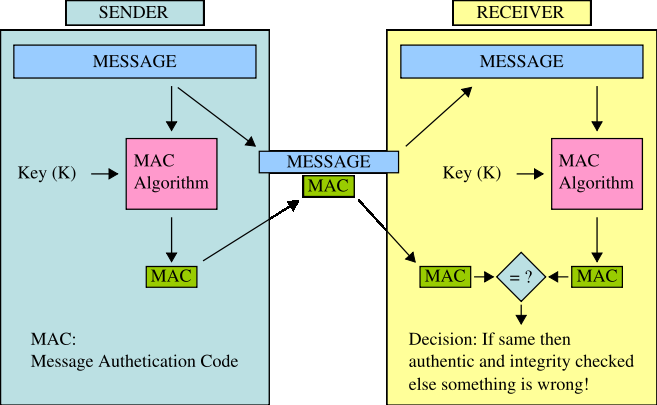
Кодом аутентификации сообщения (MAC) является короткий фрагмент информации, используемый для проверки подлинности сообщения. Алгоритм MAC принимает в качестве ввода секретный ключ и сообщение подлинности произвольной длины, и выдает MAC (иногда называют меткой). Ценность MAC в том, что защищает целостность сообщения, а также его аутентичность, позволяя контролерам (которые также обладают секретным ключом) выявлять какие-либо изменения в первоначальном содержании передаваемого сообщения.

Сообщение целостности кода (MIC), отличается от MAC в том, что секретный ключ не используется в ее деятельности. Хотя эти термины иногда используются как взаимозаменяемые, MIC всегда должен быть закодирован в ходе передачи, если он будет использоваться в качестве надежного гаранта целостности сообщения. С другой стороны, MAC, который использует секретный ключ, не обязательно должен быть зашифрован чтобы обеспечить такой же уровень надежности.

Хотя MAC функции аналогичны криптографической хэш-функции, они имеют разные требования безопасности. MAC функция должна противостоять подделке текстового сообщения. Это означает, что даже если злоумышленник имеет доступ к оракулу, который обладает секретным ключом и генерирует MAC для выбранного злоумышленником послания, то он может "никогда" угадать MAC.

MAC отличаются от цифровых подписей, ценностью MAC является одновременно получене и проверка с помощью того же секретного ключа. Это означает, что отправитель и получатель сообщения должны договориться о ключе до начала сообщения, как это имеет место в случае с симметричным шифрованием. В отличие от цифровой подписи, где используется частноый ключ из пары ключей, который является асимметричным шифрованию. Поскольку это частный ключ, доступный только для его владельца, цифровая подпись доказывает, что документ был подписан именно владельцем, а не кем-то другим. Таким образом, цифровые подписи являются гаратнтом подлиности сообщения.  
MAC алгоритмы могут быть изготовлены из других криптографических примитивов, таких, как криптографические хэш-функции (как в случае с UMAC), или для блочных алгоритмов шифрования (OMAC, CBC-MAC и PMAC).

Схема 1.Принцип MAC алгоритмов



**1.2 UHASH – универсальная функция хэширования.**

UHASH – функция хэширования – сердцевина MAC алгоритма UMAC.

Допустим функция хеширования выбирается из класса хэш-функции H, которая отображает сообщения в D, набор возможных резюме сообщения. Этот класс называется универсальным, если для каких-либо отдельных пар сообщений, имеются на множестве | H | / | D | функциий, которые отображают их в элемент D.

Это означает, что если злоумышленник хочет заменить одно сообщение другим, и, с его точки зрения, хэш-функция была выбрана абсолютно случайно, то вероятность того, что UMAC не обнаружить его изменение в большинстве случаев будет 1 / | D |.

Но это определение не является достаточно строгим, - если возможные сообщения 0 и 1, D = (0,1) и Н состоит из личности и операции «не», то H носит универсальный характер. Но если дайджест затем шифруется сложением по модулю, злоумышленник может изменить сообщение и дайджест в то же время, и приемник не распознает знать разницу.

* 1. **Математический анализ функции UHASH.**

Класс хэш-функции Н хорош в использовании тем, что затруднит для атакующего угадать правильный дайджест d фальшивого сообщение f после перехвата одного сообщения А с дайджестом С. Другими словами



должна быть очень небольшой, желательно 1 / | D |.

Можно легко построить класс хэш-функции, когда D является полем. Например, если | D | является простым, все операции, принятые по модулю | D |. Сообщение а затем кодируется как н-мерный вектор над D (a1, a2, .., аn). Н затем | D | n +1 членов, каждый из которых соответствует n +1- мерному вектору над D (h0, h1, .., hn). Если принять, что  
  
мы можем использовать правила вероятностей и комбинаторикичтобы доказать, что



Если мы правильно зашифровали все дайджесты (например, при поможи кодировки (OTP)), злоумышленник не сможет получить от них что-либо и в то же время хэш-функция может быть использована для всех контактов между двумя сторонами. Это не может быть правдой для шифрования ECB, поскольку может быть весьма вероятным, что два послания производят то же хэш-значение. Потом какой-либо вектор инициализации должен быть использован, который часто называют «nonce». Это стало распространенной практикой для установки h0 = f (nonce), где f является также секретом.

Обратите внимание, что наличие огромного количества вычислительной мощности не поможет злоумышленнику вообще. Если получатель ограничивает количество подделок она принимает ,| D | может быть 2 в 32 степени или меньше.

**2. ПОСТАНОВКА ЗАДАЧИ**

Создать хэш-функцию UMAC (message authentication code based on universal hashing).

Наша функция будет 24-битной. Причем ключ должен быть не длиннее сообщения. А зашифрованное сообщение будет также 24 бита и уже содержит ключ. Например, f («nonce»). Причем «nonce» не обязательно должно содержаться в незашифрованом сообщении.

**3. ОПИСАНИЕ ПРОГРАММЫ**

**3.1 Основные действия.**

**3.1.1 Операции со строками**

Сообщения, которые будут хэшированными рассматриваются как строки битов, заполненные нулями до определенной длины байта. После того, как сообщение мягкий, все строки рассматриваются как строки байт. А "байт" состоит из 8 бит строка. Следующие записи используется для того, чтобы манипулировать этими строками.

bytelength (S): Длина строки S в байтах.

bitlength (S): Длина строки S в битах.

zeroes(n): Строка из n нулевых байт.

S xor T: Строка, которая является результатом суммы по модулю 2 S

и Т. Строки S и T всегда имеют одинаковые

длины.

S and T: Строка, которая получается в результате побитовой коньюнкции S и T

S[i]: i-тый байт строки(индексация начинаеися с 1)

S[i...j]: Подстрока строки S состоящая из байтов i через j.

S || T: Логическое «или» строк S и T.

zeropad(S,n): Строка S заполненая нуль-битами до ближайшего положительного кратного n байту. Формально, zeropad(S,n) = S || T, где T кратчайшая строка нуль-битов (возможно пустая) ,следовательно S || T не пустое и 8n делит bitlength(S ||T).

**3.1.2 Операции с числами**

Используется тандартная запись,как для большинства математических операций, такие как "\*" для умножения, "+" для сложения и "mod" для модульного сокращения.

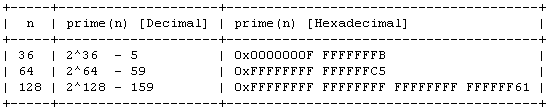
Некоторые менее стандартной нотации определяются здесь.

a^i: целое растущее до i-той мощности.

ceil(x): минимальное целое больше равное x.

начало(n): самое большое простое число менее чем 2^n.

Простые числа использованные в UMAC:



**3.1.3 Преобразовательные действия для строк и чисел.**

Преобразование между строками и целыми сделаны используя следующие функции. Каждая функция рассматривает начальные биты как более значимые чем те что идут позже

bit(S,n): Возвращает целому 1 если n-th бит строки равен S - 1, в противном . случае возвращает целое 0 (индексы начинаются с 1).

str2uint(S): Неотрицательное целое, чье бинарное представление является . трокой S. Более формально :

S is t bits long then str2uint(S) = 2^{t-1} \*

bit(S,1) + 2^{t-2} \* bit(S,2) + ... + 2^{1} \*

bit(S,t-1) + bit(S,t).

uint2str(n,i): и-тый байт строки, например str2uint(S) = n.

**3.1.4 Математические операции в строках**

Одно из первичных действий в UMAC – повторение применения операций сложения и умножения в строках. Действия "+\_32", "+\_64", и "\*\_64" определены:

"S +\_32 T" as uint2str(str2uint(S) + str2uint(T) mod 2^32, 4),

"S +\_64 T" as uint2str(str2uint(S) + str2uint(T) mod 2^64, 8), and

"S \*\_64 T" as uint2str(str2uint(S) \* str2uint(T) mod 2^64, 8).

Эти операции отлично выполняются на современных компьютерах.

**3.2 Алгоритм UHASH**

Вход:

K, строка длиной KEYLEN байт.

M, строка длиной меньше чем 2^67 бит.

taglen, числа 3,4, 8, 12 or 16.

Выход:

Y, строка длиной taglen байт.

Вычисление Y использует следующий алгоритм:

//

// одна целая итерация за 3 байта для выхода

//

iters = taglen / 3

//

// определим общее число требуемых ключей при помощи KDF.

// L1Key reuses most key material between iterations.

//

L1Key = KDF(K, 1, 1024 + (iters - 1) \* 16)

L2Key = KDF(K, 2, iters \* 24)

L3Key1 = KDF(K, 3, iters \* 64)

L3Key2 = KDF(K, 4, iters \* 4)

//

// Для каждой итерации устанавливаем свой ключ и делаем трехслойный hash.

// If bytelength(M) <= 1024, then skip L2-HASH.

//

Y = <пустая строка>

Для i = 1 to iters do

L1Key\_i = L1Key [(i-1) \* 16 + 1 ... (i-1) \* 16 + 1024]

L2Key\_i = L2Key [(i-1) \* 24 + 1 ... i \* 24]

L3Key1\_i = L3Key1[(i-1) \* 64 + 1 ... i \* 64]

L3Key2\_i = L3Key2[(i-1) \* 4 + 1 ... i \* 4]

A = L1-HASH(L1Key\_i, M)

Если (bitlength(M) <= bitlength(L1Key\_i)) тогда,

B = zeroes(8) || A

иначе получим

B = L2-HASH(L2Key\_i, A)

а если

C = L3-HASH(L3Key1\_i, L3Key2\_i, B)

Y = Y || C

end for

Return Y

**4. Стойкость алгоритма к атакам.**

В соответствии с MAC спецификой, документ является защищенным. Здесь мы описываем некоторые соображения по безопасности важные для соответствующего понимания и использования UMAC.

**4.1Сопротивление в Криптанализе**

Сила UMAC зависит от силы своих основных шифровальных функций: ключевой вывод функции (KDF) и панель-вывода функции (PDF). В этой спецификации, оба действия осуществлены используя блочный шифр, по умолчанию Передовой Шифровальный Стандарт (AES). Тем не менее, проект UMAC учитывает замену этих компонентов. На самом деле, возможно даже использовать другое блочное кодирование или другие шифровальные объекты, как например, SHA-1 или HMAC для реализации KDF или PDF.

Сердцевина проекта UMAC, функция UHASH, не зависит от шифровальных предположений: сила определена чисто математической собственностью установленной с точки зрения вероятности столкновения, и эта собственность доказывается безусловно. Это означает что сила UHASH гарантирована независимо от авансов в криптанализе.

Анализ UMAC показывает эту схему, чтобы иметь доказуемую безопасность, в смысле современной криптографии, посредством плотных уменьшений. Какие эти средства - то, что соперническая атака на UMAC, которая подделывается с вероятностью, которая значительно превышает установленную вероятность столкновения UHASH вызовет атаку сравнимой сложности. Эта атака сломает блочный шифр, в смысле отличительный блочный шифр из семейства произвольных перестановок. Этот проектый метод по существу избегает потребности в криптанализе на UMAC: как подразумевается криптоаналитические меры могли сфокусироваться в блочном шифре.

4.Повторная атака взломщика, повторяющего сообщение, nonce, и этикетку аутентификации. Во многих приложениях, посторная атака может окончательно повредить сообщение, поэтому должна иметь препятствия. В UMAC, это должно быть реализовано при получении собщения, получатель проверяет, что никакая «nonce» величина не используется дважды. При надежной связи, когда «nonce» - счетчик, это тривиально. При ненадежной связи, когда «nonce» - счетчик, в окне происходит кэширование последних «nonces». Поврежденное сообщение не запустится ,т.к. выдастса ошибка в противном случае этикетки аутентификации правильные.

Сообщение дойдет до получателя, когда данныйе (сообщение, nonce, этикетка) посчиталсись подлиными. Несомненно, этикетка должна быть в силе для сообщения и для «nonce», как определено в UMAC, но сообщение может все еще посчитаться неаутентифицированным, поскольку «nonce» будет повторен.

**5. Инструкция пользователя**

Для начала работы с программой находим umac.exe и запускаем его.

На экране появится «Enter message», после чего мы воодим нужное нам сообщение и нажимаем Enter. Затем на экране появится «Enter key», после чего мы вводим ключ (длина ключа короче чем сообщения или равна ему. В противном случае те символы, которые вышли за рамки сообщения принимать участия в кодировании не будут. ), известный только нам и получателю и снова жмем Enter.

Затем на экране видим зашифрованное сообщение. Программа выполнила требуемую операцию.

**ВЫВОДЫ**

- UMAC является самым быстрым среди MAC- алгоритмов.

- Является безопасным и криптостойким. Выдерживает множество видов атак.

-При помощи данной хэш-функции можно зашифровать скольугодно большое количество информации.

- Нет обоснования выбора конструкции, функций, констант.

-Сохраняет конфиденциальность, аутентичность, целостность,неопровергаемость.

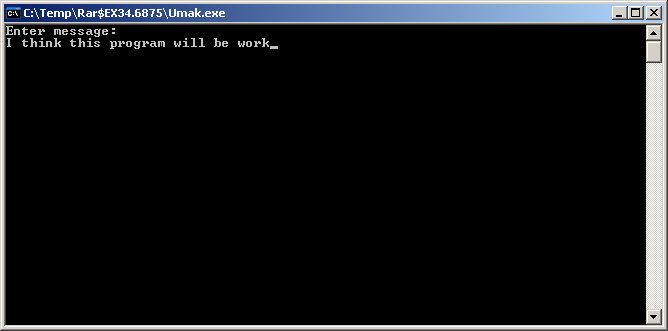
**ПЕРЕЧЕНЬ ИСПОЛЬЗОВАННОЙ ЛИТЕРАТУРЫ**

1. Зубков С. В. - Assembler – язык неограниченных возможностей. «ДМК Пресс» - 1999г.
2. http://fastcrypto.org/umac
3. http://en.wikipedia.org/wiki/UMAC
4. Кип Р. Ирвин Язык ассемблера для процессоров INTEL, 4-е изд. /Пер. с англ. – М..: – Издательский дом “ВИЛЬЯМС”, 2005 г. – 912 с., ил. – Парал. Тит. Англ.

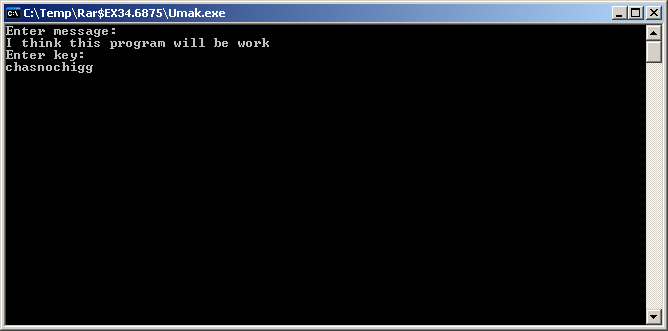
5. J. Black, S. Halevi, H. Krawczyk, T. Krovetz, and P. Rogaway, "UMAC: Fast and provably secure message authentication", Advances in Cryptology - CRYPTO '99, LNCS vol. 1666, pp. 216-233, Springer-Verlag, 1999.

Приложение А. Графическое представление программы.

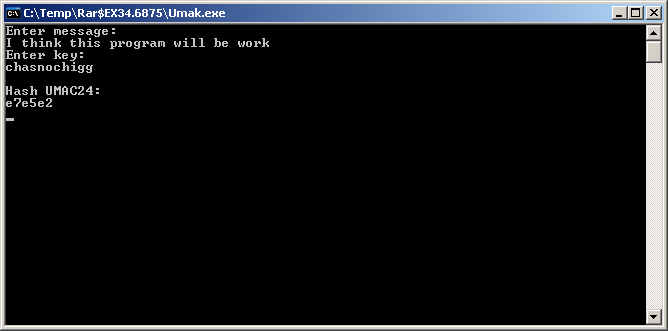
1. Открываем программу и пишем сообщение:



2. Вводим ключ:



3. Получаем зашифрованное сообщение:



**Приложение Б.**

UMAC24 - код на ассемблере.Это внешняя функция, которая прикомпилируется к коду на с++, в качестве объектного файла.

.386

.model flat,stdcall

PUBLIC UMAC24

.data

r1 db 0

r2 db 0

r3 db 0

byteCnt db 0

bitCnt db 0 ;?

counter dd 0

countmes dd 0

countres dd 0

.data?

s1 db ?

s2 db ?

s3 db ?

byte1 db ?

.code

UMAC24 proc message:BYTE, secret:BYTE, len:DWORD, result:BYTE

mov edi,len

mmm:

.if byteCnt == 0

mov ecx,counter

mov al,secret

mov bl,[eax+ecx]

mov s1,bl

inc ecx

mov bl,[eax+ecx]

mov s2,bl

inc ecx

mov bl,[eax+ecx]

mov s3,bl

inc ecx

mov counter,ecx

mov byteCnt,2

.endif

dec byteCnt

mov ecx, countmes

mov al,message

mov bl, [eax+ecx]

mov byte1,bl

inc ecx

mov countmes,ecx

mov ecx,7

metka:

mov al, byte1

AND al,1

.if ( al != 0) //msg not divisible by x

mov bl,s1 //so add s1

xor r1,bl

mov bl,s2

xor r2,bl

mov bl,s3

xor r3,bl

.endif

shr byte1,1 //divide message by x

mov al,s3

AND al,80h

.if al != 0 //and multiply secret with x, subtracting the polynomial when necessary to keep it's.

shl s3,1

mov al,s2

AND al,80h

.if al !=0

mov bl,s3

OR bl,1

mov s3,bl

shl s2,1

.endif

mov al,s1

AND al,80h

.if al !=0

mov bl,s2

OR bl,1

mov s2,bl

shl s1,1

.endif

mov al,s1

xor al,1Bh // x^24 + x^4 + x^3 + x + 1

mov s1,al

.else

shl s3,1

mov al,s2

AND al,80h

.if al !=0

mov bl,s3

Or bl,1

mov s3,bl

shl s2,1

.endif

mov al,s1

AND al,80h

.if al !=0

mov bl,s2

OR bl,1

mov s2,bl

shl s1,1

.endif

.endif

dec ecx

jns metka //for each bit in the message

dec edi

jns mmm //for each byte in the message

mov al,result

mov ecx,0

mov bl, [eax+ecx]

xor bl,r1

mov [eax+ecx],bl

inc ecx

mov bl, [eax+ecx]

xor bl,r2

mov [eax+ecx],bl

inc ecx

mov bl, [eax+ecx]

xor bl,r3

mov [eax+ecx],bl

mov bl, [eax+ecx]

xor bl,r3

mov [eax+ecx],bl

ret

UMAC24 endp

END

Это код на С++.На нем реализован ввод-вывод данных.Сам алгоритм выполняет функция.

#include <iostream.h>

#include <math.h>

#include <windows.h>

#include <stdio.h>

#include <string.h>

extern "C" \_\_stdcall UMAC24(unsigned char \*msg, unsigned char \*secret, int len, unsigned char \*result); //объявили внешнюю функцию(ту которая на асме)

int bin(bool \*str){

long b(0);

int count(0);

for(int i(31);i>=0;i--){

if(str[i]==1){b+=pow(2,count);}

count++;

}

return b;

}

void main(){

unsigned char msg[100]; //создали статические массивы размером 100 символов

unsigned char secr[100];

unsigned char rez[]={0,0,0,0}; //результат(хэш) имеет всегда фиксированное значение - 24 бита

int len(0); //длина сообщения(кол-во символов в нем)

cout<<"Enter message:"<<endl;

cin.getline(msg,100); //считываем введенное сообщение в массив msg

cout<<"Enter key:"<<endl;

cin.getline(secr,100); //считываем ключ

for(int i(0);;i++){len++;if(msg[i]==NULL){break;}} //считаем длину введенного сообщения

UMAC24(msg,secr,len,rez); //вызываем внешнюю функцию

cout<<endl<<"Hash UMAC24:"<<endl;

bool masbin[32];

int a;

for(int ii(0);ii<3;ii++){

for(int j=0;j<32;j++){

masbin[31-j]=rez[ii]%2;

rez[ii]=rez[ii]>>1;} //получили массив с представлением числа в двоичной форме

a=bin(masbin);

cout<<hex<<a; //вывод на экран в 16-ричном виде

}cout<<endl;

cin.get();

cin.get();

}