МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ
РОССИЙСКОЙ ФЕДЕРАЦИИ

Белгородский государственный технологический университет
им. В.Г. Шухова

ИНФОРМАЦИОННАЯ БЕЗОПАСНОСТЬ

Методические указания к выполнению
лабораторных работ и РГЗ для студентов очной формы
обучения направлений бакалавриата
09.03.02 - Информационные системы и технологии и
09.03.03 - Прикладная информатика

Белгород
2018

УДК 007(07)
ББК 32.81я7
И74

Составители: ст. преп. С.И. Жданова
ст. преп. С.Н. Рога

И 74 **ИНФОРМАЦИОННАЯ БЕЗОПАСНОСТЬ**: методические указания к выполнению лабораторных работ и РГЗ для студентов очной формы обучения направлений бакалавриата 09.03.02 - Информационные системы и технологии и 09.03.03 - Прикладная информатика– Белгород: Изд-во БГТУ им. В.Г. Шухова, 2018. – 82 с.

Методические указания составлены в соответствии с учебным планом и рабочей программой, предназначены для приобретения студентами базовых навыков в работе с криптографическими алгоритмами, содержат теоретический материал и задания к выполнению восьми лабораторных работ.

Методические указания предназначены для студентов очной формы обучения 09.03.02 «Информационные системы и технологии» и 09.03.03 «Прикладная информатика».

Данное издание публикуется в авторской редакции.

УДК 007(07)
ББК 32.81я7

ОГЛАВЛЕНИЕ

Правила выполнения лабораторных работ

По курсу «Информационная безопасность» предусмотрено выполнение восьми лабораторных работ и одного расчетно-графического задания (РГЗ). Студент обязан перед выполнением каждой лабораторной работы самостоятельно ознакомиться с теоретическим материалом и по результатам ее выполнения предоставить отчет. Все отчеты о выполнении лабораторных работ оформляются на листах формата А4. Каждый отчет должен содержать:

1. Заголовок лабораторной работы – номер лабораторной работы, данные о студенте, слова «Выполнение» и «Защита», название и цель работы.
2. Содержание работы и индивидуальные задания.
3. Основные понятия - краткий конспект изученного материала.
4. Ход работы – краткое описание последовательности действий, произведенных при выполнении работы. Привести текст разработанной программы и результаты ее работы. Предусмотреть ввод исходных данных как с клавиатуры, так и из текстового файла. Произвести вывод шифртекста как на экран, так и в выходной текстовый файл.
5. Вывод.

Для защиты лабораторной работы необходимо иметь распечатанный отчет и программу на электронном носителе.

Лабораторная работа №1

КЛАССИЧЕСКИЕ ШИФРЫ ПОДСТАНОВКИ

Цель работы*:* изучение классических криптографических алгоритмов моноалфавитной и многоалфавитной подстановки; криптоанализ классических шифров на основе гистограмм частот встречаемости символов алфавита.

Краткие теоретические сведения

Исходный текст - сообщение (текстовое или иной другой формы), которое требуется зашифровать.

Ключ - набор значений, используемых для шифрования исходного текста.

Зашифрованные текст - результат применения алгоритма шифрования и ключа к исходному тексту.

В шифровании симметричными ключами используется единственный ключ для кодирования и декодирования информации. Сами алгоритмы шифрования и дешифрования являются инверсными.

Введем следующие обозначения: P- исходный текст; C - зашифрованный текст; k- ключ шифрования, $E_k(x)$ -алгоритм шифрования, $D_k(x)$ - алгоритм дешифрования.

Процесс шифрования представлен формулой: $C = E_k(P)$

Процесс дешифрования представлен формулой $P = D_k(C)$, где $D_k(E_k(x))= E_k(D_k(C)) = x$.

Принципы Керкгоффса

Принципы Керкгоффса лежат в основе разработки криптографических систем. Сущность принципа заключается в том,

что чем меньше секретов содержат алгоритмы преобразования данных в криптографических системах, тем выше ее безопасность. Так, если утрата любого из секретов приводит к разрушению системы, то система с меньшим числом секретов будет надёжней. Чем больше секретов содержит система, тем более она ненадёжна и потенциально уязвима. Чем меньше секретов в системе — тем выше её прочность.

Следуя данному принципу, в системах шифрования с использованием симметричного ключа засекречивается лишь ключ; сами алгоритмы общеизвестны.

Криптоанализ

Криптоанализ - это наука о методах расшифровки секретной информации без знания ключа.

Изучение методов криптоанализа позволяет оценивать уязвимые места разрабатываемых криптографических систем. Существует 4 общих типа атак криптоанализа:

1) атака на зашифрованный текст - в распоряжении криптоаналитика имеется только зашифрованный текст. Для этой атаки применяют метод грубой силы (перехватывается зашифрованный текст и задействуются все возможные ключи, пока не получится расшифровать текст), статистическую атаку (использования статистических методов повторяемости отдельных символов в языке), атаку по образцу (повторение отдельных конструкций в зашифрованном тексте и их анализ);

2) атака со знанием исходного текста - имеются пары "зашифрованный /исходный текст". Задача криптоаналика определить алгоритм и ключ шифрования;

3) атака с выборкой исходного текста - подобна атаке со знанием исходного текста, только пары "исходный текст/зашифрованный текст" изготавливаются атакующим или лицом имеющим доступ к механизму шифрования;

4) атака с выборкой зашифрованного текста - аналогична атаке с выборкой исходного текста, только подразумевается, что есть некоторый зашифрованный текст и доступ к механизму расшифровки.

Категории классических криптографических алгоритмов

Классические шрифты бывают двух видов: шифры подстановки и шифры перестановки. Шифры подстановки подразделяются на моноалфавитные и многоалфавитные. Моноалфавитные шифры производят посимвольную замену "один к одному", многоалфавитные - для одинаковых символов исходного текста предполагается различная замена.

Моноалфавитные шифры

В моноалфавитной подстановке отношение между буквой исходного текста и буквой зашифрованного текста - один к одному.

Аддитивные шрифты

Шифр простой замены каждого символа исходного текста на символы с кодировкой сдвинутой относительно исходного текста на определенное значение. Данный подход используют в шифре **Цезаря**. Алгоритм шифрования: $C = (P+k) \bmod Z_n$, где Z_n - мощность алфавита.

Алгоритм дешифрования $P= (C-k) \bmod Z_n$.

Мультипликативные шифры

В мультипликативных шрифтах процесс шифрования подразумевает умножение исходного текста с ключом, а дешифрование - деление зашифрованного текста на ключ. Так как алгоритмы шифрования/дешифрования основаны на модульной арифметике, то дешифрование представляет собой умножение на мультипликативную инверсию ключа.

Мультипликативная инверсия ключа обозначается k^{-1} и означает, что $k \cdot k^{-1} = 1 (\mathrm{mod}\ Z_n)$

Алгоритм шифрования: $C = (P \cdot k) \bmod Z_n$, где Z_n - мощность алфавита

Алгоритм дешифрования $P = (C \cdot k^{-1}) \bmod Z_n$.

Афинный шифр

При комбинирование аддитивного и мультипликативного шрифта получается аффинный шрифт. В аффинном шрифте используется пара ключей (k_1, k_2) , где k_1 - ключ для мультипликативного шифра, k_2 - ключ аффинного шифра.

Алгоритм шифрования $C=(P \cdot k_1 + k_2) \bmod Z_n$.

Алгоритм дешифрования $P=((C-k_2) \cdot k_1^{-1}) \bmod Z_n$.

Криптоанализ шифров подстановки

Размер ключевого пространства для моноалфавитного шифра перестановки - число перестановок из n, т.е. n!. Если брать достаточное большое n , то атака грубой силы становиться довольно трудоемкой задачей. Однако моноалфавитные шифры не меняют

частоту употребления символов. Это означает, что данные типы шифров уязвимы к статистическим атакам.

Шифры подстановки. Многоалфавитные шифры

В многоалфавитной подстановке каждое появление символа может иметь различную замену. Благодаря этому изменяется частота появления символа в тексте, а значит, повышается надежность в отношении статистических атак.

Автоключевой шифр

В данном шифре ключ - это поток подключей, в котором каждый подключ зависит от позиции символа исходного текста, который используется для выбора подключа шифрования.

Алгоритм шифрования $C_i=(P_i + k_i) \mod Z_n$.

Алгоритм дешифрования $P_i=(C_i-k_i) \mod Z_n$.

Криптоанализ автоключевого шифра

Применение данного шифра позволяет скрыть частоты появления символов. Однако данный метод подвержен атаке с помощью грубой силы, так как имеет небольшую мощность множества ключей.

Шифр Плейфеера

Данный шифр использует замену биграмм. Ключ представляет собой матрицу, в которой размещены все символы исходного текста. С помощью соглашений о размещении букв в матрице можно создать множество ключей засекречивания. Шифрование проходит по паре символов исходного текста. Если две буквы пары одинаковые, то, чтобы отделить их, вставляется фиктивная буква. После вставки

фиктивных букв, если число символов в исходном тексте нечетно, в конце добавляется один дополнительный фиктивный символ, чтобы сделать число символов четным.

Шифр использует следующее правила шифрования:

1) Если буквы – пары расположены в одной строке матрицы, то соответствующий зашифрованный символ для каждой буквы – следующий символ справа в той же строке. Если символ последний в строке, то возвращаемся в начало строки.
2) Если буквы – пары располагаются в одном столбце матрицы, то соответствующий зашифрованный символ для каждой буквы - символ ниже в том же самом столбце. Если символ последний, то возвращаемся в начало столбца.
3) Если буквы – пары не располагаются в одном столбце или строке матрицы, то соответствующий зашифрованный символ для каждой буквы – символ, который находится в его собственной строке, но в том же самом столбце, что и другой символ.

Таким образом зашифрованный текст представляет собой поток ключей.

$P = P_1, P_2, P_3\ldots$

$C = C_1, C_2, C_3\ldots$

$k = [(k_1, k_2), (k_3, k_4)\ldots]$

Алгоритм шифрования $C_i = k_i$.

Алгоритм дешифрования $P_i = k_i$.

Криптоанализ шифра Плейфеера

Данный шифр не скрывает частоты появления двухбуквенных комбинаций, поэтому криптоаналитик, используя атаку на

зашифрованный текст, может провести испытания на установление частоты двухбуквенных комбинаций.

Шифр Виженера

Используется ключ длиной не более общего количества символов в алфавите. Шифрование происходит группами размером с длину ключа. Основное отличие от других многоалфавитных шифров - поток ключей зависит от позиции символа в исходном тексте, а не от него самого.

$P = P_1, P_2, P_3\ldots$

$C = C_1, C_2, C_3\ldots$

$k = [(k_1, k_2), (k_3, k_4)\ldots]$

Алгоритм шифрования $C_i = k_i$.

Алгоритм дешифрования $P_i = k_i$.

Криптоанализ шифра Виженера

Криптоанализ зашифрованного текста сводится к двум моментам.

Во-первых, необходимо подобрать длину ключа. Для этого анализируется распределение частот в зашифрованном тексте с различным прореживанием (d= 2,3...m). Анализ проводится до тех пор, пока распределение частот не станет сильно отличатся от равномерного. В этом случае предполагают, что длина ключа равняется последнему значению параметра d.

Во-вторых, текст делиться на m различных частей и к каждой части применяется атака на частоту.

Шифр Хилла

Исходный текст разбивается на блоки, в качестве ключа выступает квадратная матрица размерности соответствующей размеру блока. Требование - ключевая матрица должна иметь мультипликативную инверсию, для последующего дешифрования.

Алгоритм шифрования $C= P \cdot k \bmod Z_n$.

Алгоритм дешифрования $P= k^{-1} \cdot C \bmod Z_n$.

Криптоанализ шифра Хилла

Криптоанализ, основанный на атаке на зашифрованный текст, очень трудоемкий. В основе взлома стандартного шифра Хилла лежит уязвимость к атаке по выбранному открытому тексту. Если криптоаналитик перехватит n^2 пар символов открытого текста/зашифрованного текста, то он сможет составить систему линейных уравнений и решить ее, получив ключ.

Одноразовый блокнот

Шифр использует уникальный случайно сгенерированный ключ для каждого символа. Одноразовый блокнот - идеальный шифр, однако реализовать коммерческое применение шрифта тяжело.

Криптоанализ одноразового блокнота

Расшифровать зашифрованный текст, не имея ключа, практически невозможно.

Роторный шифр

Роторный шифр в качестве своей основ использует принципы моноалфавитной подстановки. Однако принцип имеет сложный

принцип отображения символов зашифрованного и открытого текста (см. Рис. 1).

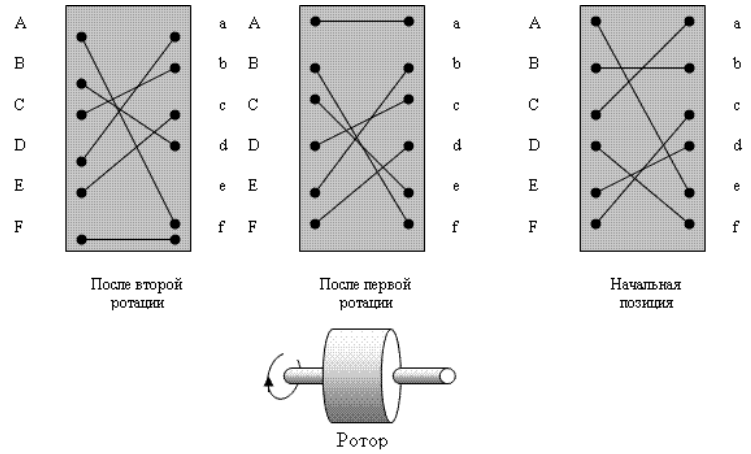


Рис. 1. Роторный шифр

Криптоанализ роторного шифра

Не сохраняет частоту употребления символов, стоек к атакам грубой силы.

Машина Энигма

Машина "Энигма" была первоначально изобретена в Сербии, но была изменена специалистами немецкой армии и интенсивно использовалась в течение Второй мировой войны. Машина базировалась на принципе шифров ротора. Рисунок 2 показывает упрощенную схему построения машины.

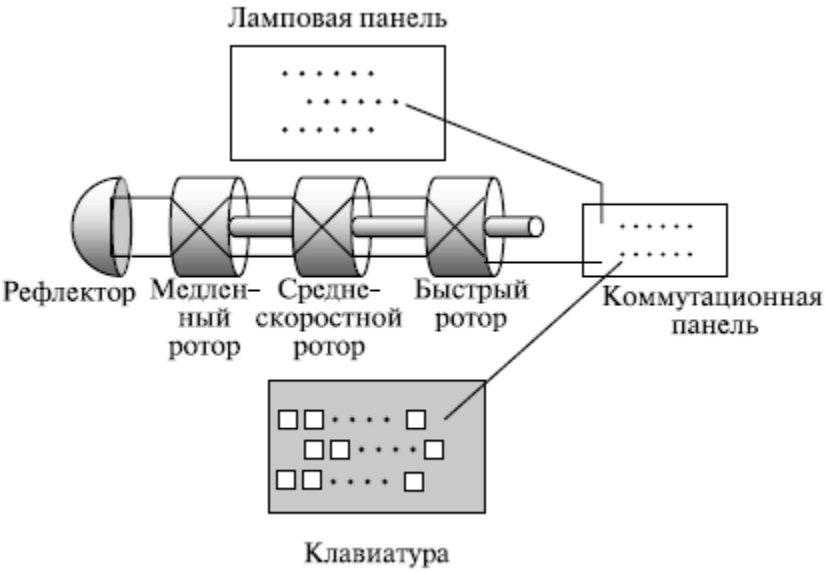


Рис. 2. Машина Энигма

Криптоанализ машины Энигма

На основании полученных копий шифровальных машин и применения статистических методов анализа и атак грубой силы машина Энигма была взломана.

Содержание работы

1) Создать программу, реализующую процесс шифрования/дешифрования текста по следующим алгоритмам:

a. аддитивный моноалфавитный шифр с задаваемым смещением
b. мультипликативный моноалфавитный шифр с задаваемым смещением
c. шифр Плейфера.

Разрабатываемая подпрограмма использует только алфавит

а б в г д е ё ж з и й к л м н о п р с т у ф х ц ч ш щ ъ ь ы э ю я _ , .

2) Провести частотный анализ символов зашифрованного текста для аддитивного и мультипликативного шифров. Вывести полученный числовые значения на экран.

3) С помощью полученной частоты встречаемости символов вручную провести и описать процесс дешифрование первых 15 символов зашифрованного сообщения.

Контрольные вопросы

1. Дать определения: моноалфавитного шифра, многоалфавитного шифра.
2. Что такое криптоанализ.
3. Перечислите методы атаки на криптографические системы.
4. Основное название криптоанализа.
5. Целесообразно ли повторное применение к зашифрованному тексту методом многоалфавитного шифрования; метода Цезаря?
6. Перечислите все 6 принципов Керкхоффа
7. Что лежит в основе статистической атаки на зашифрованный текст.
8. Дайте определение шифра Виженера.
9. Вручную расшифровать сообщение " енжят сощседэфтчфньлу рсибэдаысд й сёаэньхчжшцэхы мллайъччя анопи ". Известно, что используется шифр Виженера. Ключ "шифрование".

Таблица 1.1

Частоты встречаемости букв русского алфавита

Буква	Частота	Буква	Частота	Буква	Частота
А	0,063	**К**	0,028	**Х**	0,009
Б	0,014	**Л**	0,035	**Ц**	0,004
В	0,038	**М**	0,026	**Ч**	0,012
Г	0,013	**Н**	0,052	**Ш**	0,005
Д	0,025	**О**	0,090	**Щ**	0,003
Е	0,072	**П**	0,023	**Ъ**	0,015
Ё	0,072	**Р**	0,040	**Ы**	0,017
Ж	0,007	**С**	0,045	**Ь**	0,015
З	0,016	**Т**	0,053	**Э**	0,002
И	0,062	**У**	0,021	**Ю**	0,006
Й	0,010	**Ф**	0,001	**Я**	0,018

Таблица 1.2

Частоты биграмм русского языка

	А	Б	В	Г	Д	Е	Ж	З	И	Й	К	Л	М	Н	О	П
А	2	12	35	8	14	7	6	15	7	7	19	27	19	45	3	11
Б	5					9	1		6			6		2	21	
В	35	1	5	3	3	32		2	17		7	10	3	9	58	6
Г	7				3	3			5		1	5		1	50	
Д	25		3	1	1	29	1	1	13		1	5	1	13	22	3
Е	2	9	18	11	27	7	5	10	6	15	13	35	24	63	7	16
Ж	5	1			6	12			5					6		
З	35	1	7	1	5	3			4		2	1	2	9	9	1
И	4	6	22	5	10	21	2	23	19	11	19	21	20	32	8	13
И	1	1	4	1	3		1	2	4		5	1	2	7	9	7
К	24	1	4	1		4	1	1	26		1	4	1	2	66	2
Л	25	1	1	1	1	33	2	1	36		1	2	1	8	30	2
М	18	2	4	1	1	21	1	2	23		3	1	3	7	19	5
Н	54	1	2	3	3	34			58		3		1	24	67	2
О	1	28	84	32	47	15	7	18	12	29	19	41	38	30	9	18
П	7					15			4			9		1	46	
Р	55	1	4	4	3	37	3	1	24		3	1	3	7	56	2
С	8	1	7	1	2	25			6		40	13	3	9	27	11
Т	35	1	27	1	3	31		1	28		5	1	1	11	56	4
У	1	4	4	4	11	2	6	3	2		8	5	5	5	1	5
Ф	2					2			2						1	
Х	4	1	4	1	3	1		2	3		4	3	3	4	18	5
Ц	3					7			10		2				1	
Ч	12					23			13		2			6		
Ш	5					11			14		1	2		2	2	
Щ	3					8			6					1		
Ы		1	9	1	3	12		2	4	7	3	6	6	3	2	10
Ь		2	4	1	1	2		2	2		6		3	13	2	4
Э											1			1		
Ю		2	1	2	1			3	1		1		1	1	1	3
Я	1	3	9	1	3	3	1	5	3	2	3	3	4	6	3	6

Лабораторная работа №2

КЛАССИЧЕСКИЕ ШИФРЫ ПЕРЕСТАНОВКИ

Цель работы: изучение классических криптографических шифров перестановки и криптоанализа классических шифров перестановки на основе атак зашифрованного текста.

Краткие теоретические сведения

Шифры перестановки

Шифры перестановки основываются на изменении месторасположения символов в исходном тексте. Данные шифры не изменяют частоту букв, а это значит, что они уязвимы к частотному анализу отдельного символа.

Шифры перестановки без использования ключа

Простые шифры перестановки не используют ключа. Применяется два подхода для реализации безключевого шифра перестановки. Первый подход состоит в том, что текст записывается в таблицу столбец за столбцом, а передается по строкам. Второй способ - тест записывается в таблицу построчно, а передача производиться по столбцам. Наиболее известный - шифр изгороди (см. Рис. 3).

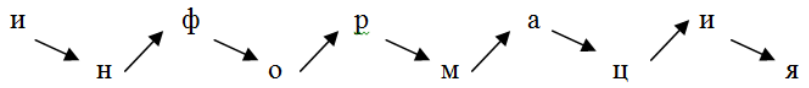


Рис. 3. Шифр изгороди.

Зашифрованное сообщение: ифраиномця.

Криптоанализ шифров перестановки без ключа

Шифр не изменяет частоту букв, поэтому для больших текстов может иметь смыл применение статистических методов исследования. Если криптоаналитику известен способ передачи и количество строк или столбцов в таблице перестановки, то можно легко расшифровать сообщение.

Ключевые шифры перестановки

Исходный текст делиться на группы определенного размера. Перестановка символов в каждой группе осуществляется по ключу.

Исходное сообщение: "Информационная безопасность"

Размер группы: 9

Ключ: 693518274

и	н	ф	о	р	м	а	ц	и	о	н	н	а	я		б	е	з	о	п	а	с	н	о	с	т	ь
1	2	3	4	5	6	7	8	9	1	2	3	4	5	6	7	8	9	1	2	3	4	5	6	7	8	9

м	и	ф	р	и	ц	н	а	о		з	н	я	о	е	н	б	а	о	ь	а	н	о	т	п	с	с
6	9	3	5	1	8	2	7	4	6	9	3	5	1	8	2	7	4	6	9	3	5	1	8	2	7	4

Зашифрованное сообщение: мифрицнао зняоенбаоьанотпсс.

Дешифрация сообщения производится использованием ключа в обратном порядке.

Криптоанализ шифров перестановки с ключом

Частота букв не меняется, поэтому целесообразно применять статистические методы исследования для больших шрифтов.

Комбинированные шифры перестановки

Комбинированные шифры перестановки включают в себя объединения двух выше рассмотренных методов. Такое сочетание позволяет достигнуть большего скремблирования (перемешивания символов).

Процесс шифрования и дешифрования состоит из следующих шагов:

1) текст пишется строка за строкой

2) делается перестановка, изменяющая порядок следования столбцов;

3) столбец за столбцом передается новая таблица.

Исходное сообщение: "Информационная безопасность"

Ключ 693518274

Шаг 1.

и	н	ф	о	р	м	а	ц	и
о	н	н	а	я		б	е	з
о	п	а	с	н	о	с	т	ь
1	2	3	4	5	6	7	8	9

Шаг 2.

м	и	ф	р	и	ц	н	а	о
	з	н	я	о	е	н	б	а
о	ь	а	н	о	т	п	с	с
6	9	3	5	1	8	2	7	4

Шаг 3.

Зашифрованное сообщение: м оизьфнаряниооцетннпабсоас.

Процесс дешифрования проходит в обратном порядке.

Криптоанализ комбинированных шифров перестановки

Шифр не изменяет частоту букв, поэтому для больших текстов может иметь смыл применение статистических методов исследования. В целом оправдано применение атаки грубой силы.

Шифры с двойной перестановкой

В шифровании дважды реализуется алгоритм перестановки. Позволяет уменьшить возможную регулярность повторения символов и заметно усложнить применение статистических методов для криптоанализа.

Криптоанализ шифров с двойной перестановкой

Возможны попытки использования статистических методов. Также подходит применение атаки грубой силы.

Содержание работы

1. Создать программу, реализующую процесс шифрования/дешифрования текста по изученным шифрам перестановки. Разрабатываемая подпрограмма использует только алфавит:

а б в г д е ё ж з и й к л м н о п р с т у ф х ц ч ш щ ъ ь ы э ю я _ , .

А Б В Г Д Е Ё Ж З И Й К Л М Н О П Р С Т У Ф Х Ц Ч Ш Щ Э Ю Я

При использовании шифра перестановки с ключом длина ключа и сам ключ должны задаваться пользователем.

Длина исходного текста должна быть не менее 10000 символов.

Подсчет количества символов должен быть осуществлен самой программой и выведен в поле для ввода.

2. Создать подпрограмму, определяющую ключ шифрования для закрытого текста, зашифрованного комбинированным шифром. Правильность подобранного ключа определяется пользователем. Если ключ подобран неверно, то пользователь отвергает его и отправляет команду на подбор следующего значения ключа. Количество символов в зашифрованном тексте никак не ограниченно.

3. Дешифровать закрытые тексты заданий своего варианта. При шифровании использовался шифр с перестановками по ключу. Дешифрование можно выполнить как вручную, так и программно.

вариант	**Шифртекст**
1	чбюЛтюолянн и з,уаосрон нтоюлбюь вю ! би оНдеепрсе тс аем.о уйдок а, лНаивсеаул нкнпв,кояю ьр лй оНыинпо од оггриврд ояе о,о к й мй еНоинти тр ыснан втзеаыен еапярьд вя еНлеешм в те нонордогтоаояет ьмнч
2	я юНло юзб л—тч а, оз юнаенма —с с пЕеетх оелйоеоодмн на,лечь л оЕсееебрвб зынке жххыьонла е , лзыРваикее ер дон бпом яырео м ; сооПлремыучпн мбтюелс клаюк ьта твге е ле в рИо,зм лодмеынпе нмзн ряоаиченто , н ь ртаВчсеп ст ьоормтаонзвх,ы до оаня ге ч,ле жоиДщраго еи начнпьелд еырхеьн в е.
3	оснтЭхо о мшалу,арааелк , внсьЛи ечля тутал, же о чсееЛур ит ыбрйсоз ипрлнлк аоб.а ювбИмолиьы алмсог о д,лк по а лыошСокнслен, цозш!л воо
4	пиЯр рлмии ссядбсьу еонйбжиезю н,о тнН еооих ,нт ыслии трые т пье внНыеиуомсмкюу коур шумне ю !

	доЖнаеа лжсоюк йур ертмее ь .
5	еВжа м е—н нпдрзауор,д лзбь ядаогро иЖ иьтк авт гуоюм оисл у Чытбоишкиортеп алоир ан Кй енрпортоуипл и
6	о Вмодт , ыоКйортрс оптоеДиклж . тАп оэаиш,ецн аоКяортмт нвё лчоаму ахннер яи т с еоВ, мд ыоКйортрс оптоеДиклж .
7	тАв оэяле саё-цпстаи,ци наи аоКяортос чтатув оерцнпушие, аоКяортмт нвё лчоаму ахннер яи т с еоВ, мд ыоКйортрс оптоеДиклж.
8	леБе петусарди оконо й туВ немаор мгоя болу..м! о Чтетищн оств нераал дойек ? о Чтнукионл к вю раднро?.ом
9	а Итгюрн—в оылт рвеещ,ствеи ааИт чмея сгтнсы рик … п и т !нУов ыаи тссчещяи н е т етИо наи тссчетяи жб !
10	оП идн мтуср явтсеел айлузи р, аН идн му лчосналцз лтоойо … Ан о,яментжйы р,псотби ру, и аК укбтд овубя рхсеьпт кой о!

Контрольные вопросы

1. Дать определения: шифр с перестановкой.
2. Какие существуют виды шифров с перестановкой.
3. В чем отличие "псевдооткрытого" текста (текст, полученный при расшифровке по ложному ключу) от настоящего.

4. В каждом из следующих шифров какое максимальное число символов будет изменено в зашифрованном тексте, если в исходном тексте изменен только один символ?

- Одиночная перестановка
- Двойная перестановка

5. Ключ шифрования в шифре перестановки — (4, 3, 6, 1, 5, 2). Найдите ключ дешифрования.

Лабораторная работа №3

Стандарт симметричного шифрования *AES*

Цель работы: реализовать процессы шифрования и дешифрования, используя алгоритм симметричного шифрования AES-128.

Краткие теоретические сведения

AES-128 - это алгоритм симметричного шифрования. Длина блока данных составляет 128 бит. Размер ключа может варьироваться: 128, 192 и 256 бит. Данный алгоритм является упрощенной версией шифра Rijndael. Основное отличие AES-128 от алгоритма Rijndael состоит в том, что последний имеет переменную длину блоков.

Используемые термины

Байт - последовательность из восьми бит. В алгоритме AES байт представляет собой элемент поля Галуа. Все операции над байтами проводятся как над элементами конечного поля GF (2^8).

Слово - последовательность из 4 байтов.

Блок - последовательность из 16 байтов. Блоки является входными и выходными данными.

Ключ - последовательность из 16, 24 и 32 байтов. Ключ шифрования является входным данным.

Матрица состояний - двумерный массив байтов, состоящий из 4-х строк. Байты в таблице располагаются в следующем порядке:

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

Раунд - итерация цикла преобразования над состоянием. Количество раундов зависит от длины ключа и может равняться от 10 до 14.

Ключ раунда - ключ, применяемый в раунде. Вычисляется для каждого раунда.

Таблица подстановок - таблица взаимно-однозначного отображения байтов.

Обратная таблица подстановок - обратное отображение таблицы подстановок. Обратная таблица подстановок и таблица подстановок связаны выражением SboxD[SboxE[a]] = a. Значения таблиц можно найти в Приложении 2.

Nb- количество слов в блоке. Nb равно длине блока, деленной на 32.

Nk - количество слов в ключе.

Nr - количество раундов.

Таблица зависимостей параметров Nr и Nk представлена ниже

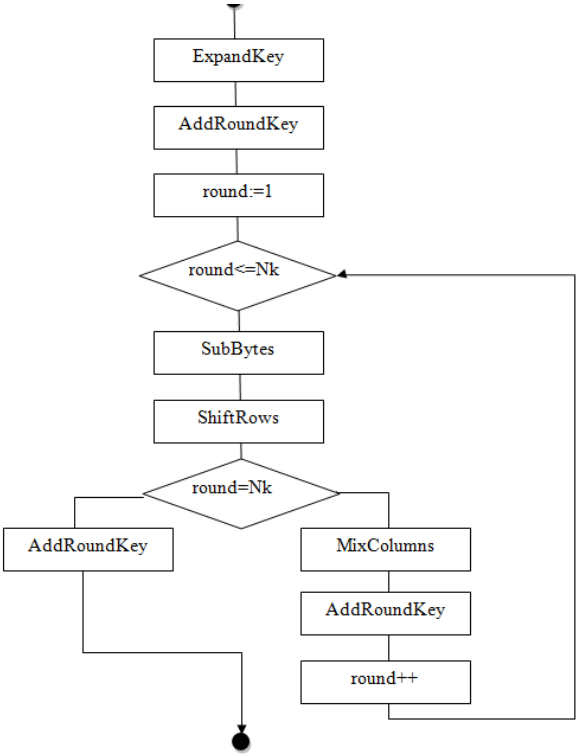
Nr	Nk
4	10
6	12
8	14

Алгоритм AES-128

Алгоритм шифрования AES-128 состоит из 5 операций преобразования:

1. ExpandKey - вычисление раундных ключей.
2. SubBytes - замена байтов исходного текста байтами таблицы подстановок.
3. ShiftRows - циклический сдвиг строк матрицы состояний на различные значения.
4. MixColumns - смешивание данных внутри каждого столбца матрицы состояния.
5. AddRoundKey - операция XOR над ключом раунда и матрицей состояний.

Блок-схема алгоритма шифрования выглядит следующим образом:



При дешифровании все процедуры производятся в обратном порядке.

Замена байт (SubBytes)

Преобразование SubBytes представляет собой нелинейную замену байт, выполняемую независимо с каждым байтом матрицы состояний.

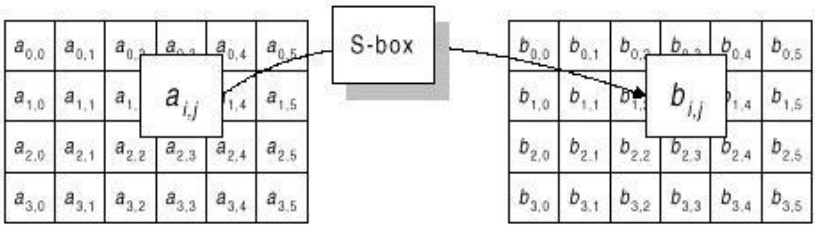


Рис. 4. Таблица подстановок

Замена происходит по таблице подстановок SboxE при шифровании и по обратной таблице подстановок SboxD при расшифровании.

Преобразование ShiftRows

Преобразование ShiftRows заключается в циклическом сдвиге влево строк матрицы состояний. Величина сдвига каждой строки представлена в таблице

Номер строки	Величина сдвига
0	0
1	1
2	2
3	3

При дешифровании первая строка матрицы состояний остается неизменной. Вторая строка сдвигается вправо на 1 байт. Третья - на 2, четвертая на 3.

Преобразование замешивания столбцов (MixColumns)

При шифровании преобразование представляет собой умножение каждого столбца матрицы состояний на квадратную матрицу 4-го порядка матрицу. Все операции производятся в поле GF (2^8)

$$\begin{bmatrix} a_{0,i}^{*} \\ a_{1,i}* \\ a_{2,i}^{*} \\ a_{3,i}^{*} \end{bmatrix} = \begin{bmatrix} 02 & 03 & 01 & 01 \\ 01 & 02 & 03 & 01 \\ 01 & 01 & 02 & 03 \\ 03 & 01 & 01 & 02 \end{bmatrix} * \begin{bmatrix} a_{0,i} \\ a_{1,i} \\ a_{2,i} \\ a_{3,i} \end{bmatrix}$$

При дешифровании берется инверсная матрица

$$\begin{bmatrix} a_{0,i}^{*} \\ a_{1,i}* \\ a_{2,i}^{*} \\ a_{3,i}^{*} \end{bmatrix} = \begin{bmatrix} 0e & 0b & 0d & 09 \\ 09 & 0e & 0b & 0d \\ 0d & 09 & 0e & 0b \\ 0b & 0d & 09 & 0e \end{bmatrix} * \begin{bmatrix} a_{0,i} \\ a_{1,i} \\ a_{2,i} \\ a_{3,i} \end{bmatrix}$$

Преобразование AddRoundKey

Выполняется операция побитового *XOR* ключа *раунда* со столбцом матрицы состояний.

$$\left[a_{0,i}^{*}, a_{1,i}*, a_{2,i}^{*}, a_{3,i}^{*}\right] = \left[a_{0,i}, a_{1,i}, a_{2,i}, a_{3,i}\right] \oplus \left[w_{round+N_b+i}\right]$$

При дешифровании ключи раунда используются в обратном порядке.

Преобразование ExpandKey

Expanded Key является линейным массивом четырехбайтных слов и обозначается как W [Nb · (Nr + 1)]. Первые Nk слов состоят из *ключа шифрования*. Остальные слова определяются *рекурсивно. Функция* расширения ключа зависит от значения Nk: существует версия функции для Nk, равным или меньшим 6, и версия для Nk больше 6.

Для Nk ≤ 6 мы имеем:

```
KeyExpansion (byte Key [4*Nk]
              word W[Nb * (Nr + 1)])
{
  for (i = 0; i < Nk; i++)
    W[i] =(Key [4*i], Key [4*i+1],
          Key [4*i+2], Key [4*i+3]);
  for (i = Nk; i < Nb * (Nr + 1); i++) {
    temp = W [i - 1];
    if (i % Nk == 0)
      temp = SubByte (RotByte (temp)) ^  Rcon [i / Nk];
    W [i] = W [i- Nk] ^ temp;
  }
}
```

В данном случае SubByte (W) является функцией, которая возвращает четырехбайтное слово, в котором каждый байт является результатом применения таблицы подстановок к байту в соответствующей позиции во входном слове. Функция RotByte (W) возвращает слово, в котором байты циклически переставлены

таким образом, что для входного слова (a, b, c, d) создается выходное слово (b, c, d, a).

Можно заметить, что первые Nk слов заполняются *ключом шифрования*. Каждое следующее слово W[i] равно XOR предыдущего слова W[i-1] и позиций слова Nk до W[i - Nk]. Для слов в позициях, которые кратны Nk, сначала применяется преобразование XOR к W[i-1] и константой раунда. Данное преобразование состоит из циклического сдвига байтов в слове RotByte, за которым следует применение табличной подстановки для всех четырех байтов в слове (SubByte).

Для Nk > 6 мы имеем:

```
KeyExpansion (byte Key [4*Nk]
              word W [Nb* (Nr+1)])
{
  for (i=0; i < Nk; i++)
    W[i]= (key [4*i], key [4*i+1],
          key [4*i+2], key [4*i+3]);
  for (i = Nk; i < Nb * (Nr + 1); i++) {
    temp = W [i-1];
    if (i % Nk == 0)
      temp = SubByte (RotByte (temp)) ^
            Rcon [i / Nk];
    else if (i % Nk == 4)
      temp = SubByte (temp);
    W[i] = W[i - Nk] ^ temp;
  }
}
```

Отличие в схеме для Nk <= 6 состоит в том, что для i-4 кратных Nk, SubByte применяется для W[i-1] перед XOR.

Цикловая константа Rcon независит от Nk и определяется следующим образом:

Раунд	**Значение Rcon**	**Раунд**	**Значение Rcon**
1	$(01\ 00\ 00\ 00)_{16}$	6	$(20\ 00\ 00\ 00)_{16}$
2	$(02\ 00\ 00\ 00)_{16}$	7	$(40\ 00\ 00\ 00)_{16}$
3	$(04\ 00\ 00\ 00)_{16}$	8	$(80\ 00\ 00\ 00)_{16}$
4	$(08\ 00\ 00\ 00)_{16}$	9	$(1B\ 00\ 00\ 00)_{16}$
5	$(10\ 00\ 00\ 00)_{16}$	10	$(36\ 00\ 00\ 00)_{16}$

При дешифровании процедура остается неизменной. Ключи раунда используются в обратном порядке.

Содержание работы

1. Изучить теоретический материал.

2. Разработать на языка C/C++ приложение, выполняющее шифрование и дешифрование файла произвольного размера на основе алгоритма AES-128.

Подробно рассмотреть действие всех цикловых преобразований, как при шифровании, так и дешифровании.

Предусмотреть ввод данных с клавиатуры как в текстовом варианте, так и в шестнадцатеричном представлении.

Ключ шифрования должен иметь переменную длину. Длина ключа задается пользователем.

3. Сохранить в отчете экранные формы, демонстрирующие процесс шифрования и дешифрования информации, проанализировать полученные результаты.

Контрольные вопросы

1. Представление байта, как элемента конечного поля Галуа $GF(2^8)$.
2. Что такое порождающий полином. Приведите примеры порождающего полинома в поле $GF(2^8)$.
3. Перечислите версии алгоритма AES и их основные отличия.
4. Опишите процесс получение ключевого расширения.
5. Докажите, что таблица подстановок и обратная таблица подстановок - инверсны:

   a) Покажите таблицу подстановок для ShiftRows. Таблица должна иметь 128 входов, но так как содержание байта не изменяется, таблица может иметь только 16 выходов; каждый выход представляет байт.

   b) Повторите часть a для обратной таблицы подстановок.

   c) Используя результаты частей a и b, докажите, что таблицы инверсны друг другу.
6. Объясните, в чем заключается дифференциальные криптоанализ алгоритма AES.
7. Объясните, в чем заключается линейный криптоанализ алгоритма AES.

Таблица подстановок

SboxE = {

0x63, 0x7C, 0x77, 0x7B, 0xF2, 0x6B, 0x6F, 0xC5, 0x30, 0x01, 0x67, 0x2B, 0xFE, 0xD7, 0xAB,0x76,

0xCA, 0x82, 0xC9, 0x7D, 0xFA, 0x59, 0x47, 0xF0, 0xAD, 0xD4, 0xA2, 0xAF, 0x9C, 0xA4, 0x72, 0xC0,

0xB7, 0xFD, 0x93, 0x26, 0x36, 0x3F, 0xF7, 0xCC, 0x34, 0xA5, 0xE5, 0xF1, 0x71, 0xD8, 0x31, 0x15,

0x04, 0xC7, 0x23, 0xC3, 0x18, 0x96, 0x05, 0x9A, 0x07, 0x12, 0x80, 0xE2, 0xEB, 0x27, 0xB2,0x75,

0x09, 0x83, 0x2C, 0x1A, 0x1B, 0x6E, 0x5A, 0xA0, 0x52, 0x3B, 0xD6, 0xB3, 0x29, 0xE3, 0x2F, 0x84, 0x53, 0xD1, 0x00, 0xED, 0x20, 0xFC, 0xB1, 0x5B, 0x6A, 0xCB, 0xBE, 0x39, 0x4A, 0x4C, 0x58, 0xCF, 0xD0, 0xEF, 0xAA, 0xFB, 0x43, 0x4D, 0x33, 0x85, 0x45, 0xF9, 0x02, 0x7F, 0x50, 0x3C, 0x9F, 0xA8, 0x51, 0xA3, 0x40, 0x8F, 0x92, 0x9D, 0x38, 0xF5, 0xBC, 0xB6, 0xDA, 0x21, 0x10, 0xFF, 0xF3, 0xD2, 0xCD, 0x0C, 0x13, 0xEC, 0x5F, 0x97, 0x44, 0x17, 0xC4, 0xA7, 0x7E, 0x3D, 0x64, 0x5D, 0x19, 0x73, 0x60, 0x81, 0x4F, 0xDC, 0x22, 0x2A, 0x90, 0x88, 0x46, 0xEE, 0xB8, 0x14, 0xDE, 0x5E, 0x0B, 0xDB, 0xE0, 0x32, 0x3A, 0x0A, 0x49, 0x06, 0x24, 0x5C, 0xC2, 0xD3, 0xAC, 0x62, 0x91, 0x95, 0xE4, 0x79, 0xE7, 0xC8, 0x37, 0x6D, 0x8D, 0xD5, 0x4E, 0xA9, 0x6C, 0x56, 0xF4, 0xEA, 0x65, 0x7A, 0xAE, 0x08,

0xBA, 0x78, 0x25, 0x2E, 0x1C, 0xA6, 0xB4, 0xC6, 0xE8, 0xDD, 0x74, 0x1F, 0x4B, 0xBD, 0x8B,0x8A,

0x70, 0x3E, 0xB5, 0x66, 0x48, 0x03, 0xF6, 0x0E, 0x61, 0x35, 0x57, 0xB9, 0x86, 0xC1, 0x1D, 0x9E,

0xE1, 0xF8, 0x98, 0x11, 0x69, 0xD9, 0x8E, 0x94, 0x9B, 0x1E, 0x87, 0xE9, 0xCE, 0x55, 0x28, 0xDF,

0x8C, 0xA1, 0x89, 0x0D, 0xBF, 0xE6, 0x42, 0x68, 0x41, 0x99, 0x2D, 0x0F, 0xB0, 0x54, 0xBB, 0x16

};

Обратная таблица подстановок

SboxD = {

0x52, 0x09, 0x6A, 0xD5, 0x30, 0x36, 0xA5, 0x38, 0xBF, 0x40, 0xA3, 0x9E, 0x81, 0xF3, 0xD7, 0xFB,

0x7C, 0xE3, 0x39, 0x82, 0x9B, 0x2F, 0xFF, 0x87, 0x34, 0x8E, 0x43, 0x44, 0xC4, 0xDE, 0xE9, 0xCB,

0x54, 0x7B, 0x94, 0x32, 0xA6, 0xC2, 0x23, 0x3D, 0xEE, 0x4C, 0x95, 0x0B, 0x42, 0xFA, 0xC3, 0x4E,

0x08, 0x2E, 0xA1, 0x66, 0x28, 0xD9, 0x24, 0xB2, 0x76, 0x5B, 0xA2, 0x49, 0x6D, 0x8B, 0xD1,0x25,

0x72, 0xF8, 0xF6, 0x64, 0x86, 0x68, 0x98, 0x16, 0xD4, 0xA4, 0x5C, 0xCC, 0x5D, 0x65, 0xB6,0x92,

0x6C, 0x70, 0x48, 0x50, 0xFD, 0xED, 0xB9, 0xDA, 0x5E, 0x15, 0x46, 0x57, 0xA7, 0x8D, 0x9D, 0x84,

0x90, 0xD8, 0xAB, 0x00, 0x8C, 0xBC, 0xD3, 0x0A, 0xF7, 0xE4, 0x58, 0x05, 0xB8, 0xB3, 0x45,0x06,

0xD0, 0x2C, 0x1E, 0x8F, 0xCA, 0x3F, 0x0F, 0x02, 0xC1, 0xAF, 0xBD, 0x03, 0x01, 0x13, 0x8A, 0x6B,

0x3A, 0x91, 0x11, 0x41, 0x4F, 0x67, 0xDC, 0xEA, 0x97, 0xF2, 0xCF, 0xCE, 0xF0, 0xB4, 0xE6, 0x73,

0x96, 0xAC, 0x74, 0x22, 0xE7, 0xAD, 0x35, 0x85, 0xE2, 0xF9, 0x37, 0xE8, 0x1C, 0x75, 0xDF, 0x6E,

0x47, 0xF1, 0x1A, 0x71, 0x1D, 0x29, 0xC5, 0x89, 0x6F, 0xB7, 0x62, 0x0E, 0xAA, 0x18, 0xBE, 0x1B,

0xFC, 0x56, 0x3E, 0x4B, 0xC6, 0xD2, 0x79, 0x20, 0x9A, 0xDB, 0xC0, 0xFE, 0x78, 0xCD, 0x5A, 0xF4,

0x1F, 0xDD, 0xA8, 0x33, 0x88, 0x07, 0xC7, 0x31, 0xB1, 0x12, 0x10, 0x59, 0x27, 0x80, 0xEC, 0x5F,

0x60, 0x51, 0x7F, 0xA9, 0x19, 0xB5, 0x4A, 0x0D, 0x2D, 0xE5, 0x7A, 0x9F, 0x93, 0xC9, 0x9C, 0xEF,

0xA0, 0xE0, 0x3B, 0x4D, 0xAE, 0x2A, 0xF5, 0xB0, 0xC8, 0xEB, 0xBB, 0x3C, 0x83, 0x53, 0x99, 0x61,

0x17, 0x2B, 0x04, 0x7E, 0xBA, 0x77, 0xD6, 0x26, 0xE1, 0x69, 0x14, 0x63, 0x55, 0x21, 0x0C, 0x7D

};

Лабораторная работа №4

ГЕНЕРАЦИЯ БОЛЬШИХ ПРОСТЫХ ЧИСЕЛ И АЛГОРИТМЫ ТЕСТИРОВАНИЯ НА ПРОСТОТУ

Цель работы: освоить методы генерации больших простых чисел, изучить и реализовать вероятностные и детерминированные алгоритмы проверки чисел на простоту.

Краткие теоретические сведения

Генерация случайных чисел

Генерация случайных чисел является неотъемлемой частью многих криптографических алгоритмов. Введение случайной величины в процессе шифрования позволяет обеспечить надежное управление ключевой информацией.

Обычно при создании последовательности случайных чисел предполагается, что данная последовательность чисел должна быть случайной в некотором определенном статистическом смысле. Следующие два критерия используются для доказательства того, что последовательность чисел является случайной:

1. **Однородное распределение**: распределение чисел в последовательности должно быть однородным; это означает, что частота появления каждого числа должна быть приблизительно одинаковой.

2. **Независимость**: ни одно значение в последовательности не должно зависеть от других.

В приложениях, таких как взаимная аутентификация и генерация ключа сессии члены последовательности должны быть

непредсказуемы. При "правильной" случайной последовательности каждое число статистически не зависит от остальных чисел и, следовательно, непредсказуемо. Однако правильные случайные числа на практике используются достаточно редко, чаще последовательность чисел, которая должна быть случайной, создается некоторым алгоритмом. В данном случае необходимо, чтобы противник не мог предугадать следующие элементы последовательности, основываясь на знании предыдущих элементов и используемого алгоритма.

Генерация больших простых чисел

Простое число – это такое число, которое не имеет других делителей кроме себя самого и единицы.

Любое целое число может быть разложено на множители и единственным способом представлено в виде:

$$a = p_1^{\alpha_1} \cdot p_2^{\alpha_2} \cdot \ldots \cdot p_n^{\alpha_n}$$ где p_i - простые числа.

Наибольший общий делитель $НОД(a,b) = \max(k : \mathrm{k} \mid \mathrm{a} \text{ и } \mathrm{k} \mid \mathrm{b})$ где $k|a$ обозначает, что k делит a без остатка.

Существует приблизительно 10^{151} простых чисел длиною от 1 бита до 512 включительно. Для чисел, близких к n, вероятность того, что выбранное число окажется простым, равна 1/ln(n). Поэтому полное число простых чисел, меньших n, равно n/ln(n). Считается, что вероятность выбора двумя людьми одного и того же большого простого числа очень мала.

Тестирование на простоту

Выделяют два типа критериев простоты: детерминированные и вероятностные. Детерминированные тесты позволяют доказать, что

введенное число - простое. Вероятностные тесты используются для тестирования отдельных чисел на простоту, однако их результат, с некоторой вероятностью, может быть неверен. Для снижения верности ошибки увеличивают количество повторений теста с модифицированными исходными данными. Для проверки чисел на простоту, как правило, используют либо комбинацию вероятностного и детерминированного теста либо же только детерминированный тест.

Вероятностные тесты на простоту

Для того, чтобы проверить, является ли число n простым, выбирают случайное число a, такое что $1<a<n$.

Если число n не проходит тест по основанию a, то говорят "число n составное".

Если число n проходит тест по основанию a, то однозначно сказать о его простоте нельзя. Число n тестируется m раз по разным a и если во всех случаях тест дает ответ " число n, вероятно, простое", то можно утверждать, что n является простым с вероятностью близкой к 1.

Примерное количество итераций цикла определяется формулой $m = \log_p (1-p_0)$ где p - вероятность того, что число пройдет тест, p_0 - необходимая вероятность того, что число является простым.

Арифметика вычетов

Будем рассматривать целые числа в связи с остатками от деления на натуральное число m, называемое модулем. Если 2 целых числа a и b имеют одинаковые остатки от деления на m, то они называются ***сравнимыми по модулю m***: $a \equiv b(\bmod m)$, где $a = k \cdot m + b$ для некоторого целого k.

Иногда b называют ***вычетом a по модулю m*** , a – ***конгруэнтным*** b ***по модулю*** m. Операция c *mod* t означает остаток от деления с на t и называется ***приведением по модулю***.

Числа, сравнимые по модулю, образуют класс вычетов по модулю m. Все числа из одного и того же класса имеют один и тот же остаток r от деления на m. Любое число a из класса вычетов называется вычетом по модулю m.

Полная система вычетов – это совокупность вычетов, взятых по одному из каждого класса вычетов.

Приведённая система вычетов по модулю m – это совокупность всех вычетов из полной системы, взаимно простых с модулем m.

Теоремы Ферма и Эйлера

Функция Эйлера $\varphi(m)$– это количество положительных целых чисел, которые меньше m и взаимно простые с m.

Например, $\varphi(10)=4$, т.к. числа 2, 4, 6 и 8 имеют общий с 10 делитель 2, число 5 имеет общий с 10 делитель 5, а числа 1, 3, 7 и 9 – не имеют общих делителей с 10 (кроме 1) и, следовательно, являются взаимно простыми с числом 10.

Если p – простое, то $\varphi(p)=p-1$

Если p ,q – простые, то

$$\varphi(n)= \varphi(p\cdot q)= \varphi(p)\cdot\varphi(q)=(p-1)\cdot(q-1)$$

$$\varphi(n)= \varphi(p^{\alpha}\cdot q^{\beta})= \varphi(p^{\alpha})\cdot\varphi(q\beta)=(p-1)\cdot p^{\alpha-1}\cdot(q-1)\cdot q^{\beta-1}$$

Теорема Эйлера: Для любых взаимно простых чисел a и m справедливо $a^{\varphi(n)} \equiv 1 \pmod m$.

Следствие из теоремы Эйлера: $a^{\varphi(m)+1} \equiv a \pmod m$

Теорема Ферма: Если p– простое число, a– положительное целое число, которое не делится на p, то $a^{p-1} \equiv 1 \pmod p$).

Тест Ферма

Из малой теоремы Ферма следует, что, если для нечетного n существует такое целое a, что $1<=a<=n$, НОД $(a,n)=1$ и $a^{n-1} \equiv 1 (\mathrm{mod}\ p)$, то число n вероятно простое.

Входные данные: нечетное число $n \geq 5$;

Выходные данные: "число n составное", " число n вероятно простое".

1. Выбирается случайное целое число a, $2<=a<=n-2$.
2. Вычисляется $r = a^{n-1} (\mathrm{mod}\ n)$.
3. При $r=1$ тест дает ответ, что "число n вероятно простое". Иначе " число n составное ".

Существенным недостатком данного теста является наличие чисел Кармайкла. Это нечетные составные числа, для которых сравнение из формулы выполняются при любом a.

Тест Рабина-Миллера

Тест Рабина-Миллера чаще всего используется для проверки числа на простоту.

Входные данные: нечетное число $n>=5$;

Выходные данные: "число n составное", "число n вероятно простое ".

1. Представить $n-1$ в виде $n-1=2^s$, число r - нечетное.
2. Выбирается случайное целое число a, $2<=a<=n-2$.
3. Вычислить $y = a^r (\mathrm{mod}\ n)$
4. При $y \neq 1$ и $y \neq n-1$ выполнить следующие действия:

   4.1 Пусть j=1

   4.2 Если j<=s-1 и y≠n-1

4.2.1 Положить $y=y^2 \pmod n$

4.2.2 При y=1 результат: " число *n* составное "

4.2.3 Увеличить j на 1

4.3 При $y \neq n-1$ результат: " число *n* составное ".

5. Результат: " число n вероятно простое ".

В результате выполнения теста для простого числа *n* единственными решениями сравнения $y^2 \equiv 1 \pmod n$ являются $y = \pm 1 \pmod n$.

Вероятность ошибки теста Рабина - Миллера составляет не более 1/4. Для уменьшения вероятности ошибки тест необходимо повторить не менее *t* раз. Вероятность ошибки в этом случае будет равняться $(1/4)^t$.

Детерминированные алгоритмы проверки на простоту

Детерминированный алгоритм, проверяющий простоту чисел, принимает целое число и выдает на выходе признак: это число — простое число или составной объект.

Алгоритм теории делимости

Испытание на делимость - самое простое детерминированное испытание. В его основе лежит использование в качестве делителей всех чисел меньших, чем $\sqrt{n}$. Если любое из этих чисел делит *n*, то *n* - составное число.

Сложность побитового испытания делимостью

Пусть дано число n. Разрядность числа - 200 бит. Вычислим количество разрядных операций, необходимых для выполнения алгоритма - $2^{\frac{n_b}{2}} = 2^{100}$.

Если алгоритм имеет скорость 2^{30} операций в секунду, то необходимо 2^{70} секунд для проведения испытаний.

AKS-алгоритм

В 2002 году летом индийские математики Аграавал, Саксен и Кайал нашли полиномиальный детерминированный алгоритм проверки числа на простоту. Основная идея, на которой основан это алгоритм такова: n- простое число тогда и только тогда, когда:

$$\text{НОД}(a,n)=1,\ (x-\underline{a})^n \equiv (x-a)^n \pmod{n}$$

Алгоритм AKS (псевдокод)

```
if (n is has the form a^b with b > 1) then output COMPOSITE
r := 2
while (r < n) {
        if (gcd(n,r) is not 1) then output COMPOSITE
        if (r is prime greater than 2) then {
                let q be the largest factor of r-1
                if (q > 4sqrt(r)log n) and (n^(r-1)/q is not 1 (mod r)) then
break
        }
        r := r+1
}
for a = 1 to 2sqrt(r)log n {
```

if ($(x-a)^n$ is not (x^n-a) (mod x^r-1,n)) then output COMPOSITE

}

output PRIME;

Алгоритм AKS и его модификации на практике пока не могут конкурировать по удобству использования и времени выполнения с другими алгоритмами.

Реализацию алгоритма AKS можно посмотреть по приведенной ниже ссылке:

http://yves.gallot.pagesperso-orange.fr/src/

Алгоритм генерации простого числа

1. Сгенерировать случайное *n*-битное число p.
2. Установить его старший и младший биты равными 1. Старший бит будет гарантировать требуемую длину искомого числа, а младший бит – обеспечивать его нечётность.
3. Убедиться, что p не делится на небольшие простые числа: 3, 5, 7, 11 и т.д. Наиболее эффективна проверка на делимость на все простые числа, меньшие 500.
4. Выполнить тест Рабина-Миллера минимум 5 раз.

Если *p* не прошло хотя бы одну проверку в пунктах 3 или 4, то оно не является простым.

Проверка, что случайное нечётное *p* не делится на 3, 5 и 7, отсекает 54 % нечётных чисел. Проверка делимости на все простые числа, меньшие 256, отсекает 80 % составных нечётных чисел.

Даже если составное число «просочилось» через этот алгоритм, это будет сразу же замечено, т.к. шифрование и дешифрование не будут работать.

Содержание работы

Реализовать приложение, позволяющее выполнять следующие действия:

1. Реализовать подпрограмму для проверки чисел на простоту, используя изученные вероятностные методы.

Программа должна отражать затраченное время на проверку чисел на простоты.

В тесте Ферма предусмотреть вывод сообщения о введении числа Кармайкла, если такое число подается на проверку.

В тесте Рабина-Миллера пользователь должен иметь возможность самостоятельно задавать количество проверок.

2. Реализовать подпрограмму для проверки чисел на простоту, используя предложенный детерминированный алгоритм(алгоритм теории делимости).

Программа должна отражать затраченное время на проверку чисел на простоты.

3. С помощью алгоритма генерации простого числа получить большое простое число (в данной лабораторной работе под большими числами будем понимать числа, превышающие 2^{64}).

Пользователь вводит количество проверок в тесте на простоту и длину числа в битах.

Контрольные вопросы

1. Для чего нужно большое простое число?
2. Как проверить, является число простым или нет?
3. Как сгенерировать большое простое число?

4. Приведите пример алгоритма эффективной реализации возведения целого числа в целую степень по модулю n.

5. Что такое псевдопростые числа? Как они влияют на результат работы алгоритмов проверки на простоту.

6. Что такое числа Кармайкла? Перечислите числа Кармайкла в диапазоне от 1 до 100000.

Лабораторная работа №5

АЛГОРИТМ ОБМЕНА КЛЮЧАМИ ДИФФИ-ХЕЛЛМАНА

Цель работы: изучить и реализовать на практике алгоритм обмена ключами Диффи-Хеллмана.

Краткие теоретические сведения

Криптографический протокол – это набор формализованных правил, описывающий последовательность действий, исполняемые двумя и более лицами, для решения задачи защиты информации с использованием криптографии.

Одна из основных проблем криптографии – передача сообщения по незащищенному каналу связи. При симметричном шифровании ключи шифрования ценны также как и само передаваемое сообщение. Отсюда следует проблема распределения ключей. Надежность любой симметричной криптографической системы во многом зависит от выбранной схемы распределения ключей.

Первообразные корни

Класс $[a]$, где $(a,n)=1$ называется первообразным корнем по модулю n, если показатель числа a по модулю n равен $\varphi(n)$ значению функции Эйлера.

Известно, что любой показатель k числа a по модулю n делит $\varphi(n)$. Из этого факта вытекает способ нахождения первообразных корней. Представим $\varphi(n)$ в виде: $\varphi(n) = p_1^{k_1} \cdot p_2^{k_2} \ldots \cdot p_m^{k_m}$, где p_i - простые делители. Тогда, чтобы k было первообразным корнем числа n, нужно, чтобы:

1) число k было взаимно простым с числом n ;

2) $k^{\frac{\varphi(n)}{p_i}} \neq 1 \pmod{n} \; \forall i$

Построение первообразного корня по модулю *n*

В силу теоремы Эйлера для любых взаимно простых a и n выполняется соотношение: $a^{\varphi(n)} \equiv 1 \pmod{n}$, где $\varphi(n)$ – обозначает ***функцию Эйлера***, значение которой равно количеству положительных целых значений, меньших n и взаимно простых с n .

Для простого числа n выполняется: $\varphi(n) = n - 1$

Если предположить, что два числа p и q – простые, тогда для $n = p^{\alpha} \cdot q^{\beta}$ функция Эйлера будет иметь вид:

$$\varphi(n) = \varphi(p^{\alpha} \cdot q^{\beta}) = \varphi(p^{\alpha}) \cdot \varphi(q^{\beta}) = (p-1) \cdot p^{\alpha-1} \cdot (q-1) \cdot q^{\beta-1}.$$

Говорят, что число a, взаимно простое с модулем n, ***принадлежит показателю m*** , если m – такое наименьшее натуральное число, что выполняется сравнение: $a^m \equiv 1 \pmod{n}$

Если a и n – взаимно простые, то существует, по крайней мере, одно число $m = \varphi(n)$, удовлетворяющее условию $a^m \equiv 1 \pmod{n}$.

Если некоторая последовательность имеет длину ϕ(n), тогда целое число a генерирует своими степенями множество всех ненулевых вычетов по модулю n. Такое целое число называют Для числа n количество ***первообразных корней по модулю n*** равно $\varphi(\varphi(n-1))$.

Пример. Пусть n = 41 . Имеем c= φ (41)= 40=$2^3 \cdot 5$. Итак, первообразный корень не должен удовлетворять двум сравнениям: $a^8 \equiv 1 \pmod{41}$, $a^{20} \equiv 1 \pmod{41}$.

Испытываем числа 2, 3, 4, … : $2^8 \equiv 10$, $2^{20} \equiv 1$, $3^8 \equiv 1$, $3^{20} \equiv 40$, $4^8 \equiv 18$, $4^{20} \equiv 1$,$5^8 \equiv 18$, $5^{20} \equiv 1$, $6^8 \equiv 10$, $6^{20} \equiv 40$. Отсюда видно, что 6 – наименьший первообразный корень по модулю 41.

Распределение секретных ключей. Схема обмена ключами Диффи-Хеллмана

Цель схемы Диффи-Хелмана – обеспечение двум пользователям защищенную возможность обмена секретным ключом. Алгоритм основан на трудности вычислений дискретных логарифмов.

Безопасность обмена ключами в алгоритме Диффи-Хеллмана вытекает из того факта, что, хотя относительно легко вычислить экспоненты по модулю простого числа, но очень трудно вычислить дискретные логарифмы. Для больших простых чисел задача считается неразрешимой.

Предположим, что двум абонентам необходимо провести конфиденциальную переписку, а в их распоряжении нет первоначально оговоренного секретного ключа. Однако между ними существует канал, защищённый от модификации, т.е. данные, передаваемые по нему, могут быть прослушаны, но не изменены (такие условия имеют место довольно часто). В этом случае две стороны могут создать одинаковый секретный ключ, ни разу не передав его по сети, по следующему алгоритму (см. рисунок 5).

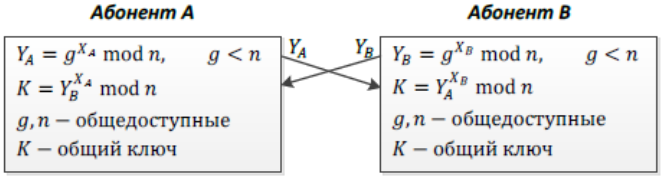


Рис. 5. Обмен ключами по схеме Диффи-Хеллмана

Алгоритм заключается в следующем:

Задаются глобальные открытые элементы:

- n– случайное большое простое число;
- g– первообразный корень n.

Вычисляется ключ пользователем A:

- выбирается большое секретное число X_A (X_A <n);
- вычисление открытого значения Y_A : $Y_A = g^{X_A} \bmod n$;

Вычисляется ключ пользователем B:

- Выбирается большое секретное число X_B (X_B <n);
- Вычисление открытого значения Y_B: $Y_B = g^{X_B} \bmod n$;

Вычисляется секретный ключ пользователем A: $K=Y_B^{X_A} \bmod n$;

Вычисляется секретный ключ пользователем B: $K=Y_A^{X_B} \bmod n$.

Данный алгоритм надежно работает только на каналах, защищенных от модификации.

С принципом работы алгоритма Диффи-Хелмана можно познакомиться в следующем видео: https://habrahabr.ru/post/151599/.

Содержание работы

Реализовать приложение, позволяющее выполнять следующие действия:

1. Изучить схему обмена ключами Диффи-Хелмана.
2. Реализовать подпрограмму, определяющую для заданного числа первые 100 первообразных корней, отображая при этом суммарное время, затраченное на их поиск. Число может задаваться десятеричной, шестнадцатеричной и двоичной формах.
3. Вручную для первых 5 полученных числовых значений привести доказательство, что они действительно являются первообразными корнями заданного числа n.

4. Реализовать подпрограмму, моделирующую обмен ключами между абонентами по схеме Диффи-Хеллмана. Программа должна получать большие простые числа X_A, X_B и n случайным образом с помощью алгоритма генерации простого числа, а также предоставлять пользователю возможность задавать их.

Контрольные вопросы

1. Что такое первообразный корень по модулю n.
2. Для числа n=54 найти первообразные корни.
3. Опишите алгоритм эффективной реализации возведения целого числа в целую степень по модулю n.
4. Опишите схему обмена ключами Диффи-Хелмана.
5. В чем заключается криптографическая стойкость алгоритма обмена ключами Диффи-Хелмана.
6. Доказать, что в схеме Диффи-Хелмана $K_a=K_b$.

Лабораторная работа №6

РЕАЛИЗАЦИЯ КОМБИНИРОВАННЫХ АЛГОРИТМОВ ШИФРОВАНИЯ ДАННЫХ

Цель работы: изучить принцип работы ассиметричных алгоритмов. Разработать приложение, совмещающее в себе достоинства симметричных и ассиметричных алгоритмов шифрования.

Краткие теоретические сведения

При всех своих достоинствах симметричные алгоритмы обладают существенными недостатками. Во-первых, при симметричном шифровании необходимо, чтобы приемник и передатчик сообщения имели общий **ключ**, который каким-то образом должен быть им заранее передан. Один из основоположников шифрования с открытым ключом У. Диффи, заметил, что данное требование отрицает всю суть криптографии, а именно возможность поддерживать всеобщую секретность при коммуникациях.

Во-вторых, все участники обмена данными должны быть убеждены, что электронное сообщение было послано конкретным участником. Это более сильное требование, чем **аутентификация**.

Диффи и Хеллман достигли значительных результатов, предложив способ решения обеих задач, который радикально отличается от всех предыдущих подходов к шифрованию.

Ключ, используемый в симметричном шифровании, будем называть **секретным ключом**. Два ключа, используемые при шифровании с открытым ключом, будем называть **открытым ключом** и **закрытым ключом**. Закрытый ключ держится в секрете, создается

локально каждым пользователем, открытый ключ – общеизвестен. Закрытый ключ обозначается KR, открытый ключ -KU.

Диффи и Хеллман описывают требования, которым должен удовлетворять алгоритм шифрования с открытым ключом.

1) Создавать пару (открытый ключ KU, закрытый ключ KR).

2) Имея открытый ключ и незашифрованное сообщение M, создать соответствующее зашифрованное сообщение:

3) $C = E_{KU}[M]$

4) Дешифровать сообщение, используя закрытый ключ:

5) $M = D_{KR}[C] = D_{KR}[E_{KU}[M]]$

6) Невозможно, зная *открытый ключ* KU, определить закрытый ключ KR.

7) Невозможно, зная *открытый ключ* KU и зашифрованное сообщение C, восстановить исходное сообщение M.

8) Можно добавить шестое требование, хотя оно не выполняется для всех алгоритмов с открытым ключом:

9) Шифрующие и дешифрующие функции могут применяться в любом порядке:

10) $M = E_{KU}[D_{KR}[M]]$

Односторонней функцией называется такая функция, у которой каждый аргумент имеет единственное обратное значение, при этом вычислить саму функцию легко, а вычислить обратную функцию трудно.

Обычно "легко" означает, что проблема может быть решена за полиномиальное время от длины входа. Таким образом, если *длина* входа имеет *n* битов, то время вычисления функции пропорционально n^a, где *a* - фиксированная константа. Таким образом, говорят, что *алгоритм* принадлежит классу полиномиальных алгоритмов P. Термин

"трудно" означает более сложное понятие. В общем случае будем считать, что проблему решить невозможно, если усилия для ее решения больше полиномиального времени от величины входа. Например, если длина входа *n* битов, и время вычисления функции пропорционально 2^n, то это считается вычислительно невозможной задачей.

Ассиметричное шифрование

Схема шифрования

Шифрование с открытым ключом состоит из следующих шагов:

1) Пользователь B создает пару ключей KU_b и KR_b, используемых для шифрования и *дешифрования* передаваемых сообщений.

2) Пользователь B делает доступным некоторым надежным способом свой ключ шифрования, т.е. открытый ключ KU_b. Составляющий пару закрытый ключ KR_b держится в секрете.

3) Если A хочет послать сообщение B, он шифрует сообщение, используя открытый ключ B KU_b.

4) Когда B получает сообщение, он дешифрует его, используя свой закрытый ключ KR_b. Никто другой не сможет дешифровать сообщение, так как этот закрытый ключ знает только B.

Достоинства и недостатки асимметричных алгоритмов.

Достоинства:

- отсутствие необходимости передачи секретного ключа;
- число ключей в ассиметричной системе меньше, чем в симметричной;

Недостатки:

- низкое быстродействие;
- используются более длинные ключи, чем симметричные;
- требуются большие вычислительные ресурсы.

Алгоритм шифрования RSA

Алгоритм RSA был одним из первых алгоритмов ассиметричного шифрования. Он был разработан в 1977 году Роном Ривестом, Ади Шамиром и Леном Адлеманом. С тех пор алгоритм Rivest-Shamir-Adleman (*RSA*) широко применяется практически во всех приложениях, использующих криптографию с открытым ключом.

Алгоритм основан на использовании того факта, что задача **факторизации** является трудной, т.е. легко перемножить два числа, в то время как не существует полиномиального алгоритма нахождения простых сомножителей большого числа.

Алгоритм ***RSA*** представляет собой блочный *алгоритм шифрования*, где зашифрованные и незашифрованные данные являются целыми числами между 0 и n -1 для некоторого n.

Вычисление ключей

Для алгоритма RSA этап создания ключей состоит из следующих операций**:**

1) Выбираются два простых различных числа p и q. Вычисляется n= p*q

2) Вычисляется функция Эйлера φ(n)=(p-1)·(q-1)

3) Выбирается произвольное число *e*, такое что $1<e<n$ и *НОД (n,e)=1*

4) Вычисляется d , такое что 1<d<n. Число d должно удовлетворять условию $e \cdot d \bmod \varphi(n) = 1$. Для нахождения d используется расширенный алгоритм Евклида. Число d является частью закрытого ключа и хранится в секрете.

5) Открытым ключом будем называть пару числе KU (e,n).

6) Закрытый ключ KR (d,n).

Расширенный алгоритм Евклида

Расширенный алгоритм Евклида состоит из двух частей. Первая часть позволяет вычислить НОД двух чисел. При вычислении используется рекуррентное правило:

НОД (A,B)= НОД (B,A mod B)

Вычисление производятся до тех пор, пока A mod B = 0. Последнее ненулевое значение является искомым делителем.

Вторая часть алгоритма позволяет решить уравнение вида $A * x + B * y = d$, где A, B - заданные числа, d- НОД (A,B).

В расширенном алгоритме Евклида, помимо рекуррентных соотношений из первой части, используются также и следующие соотношения:

$$x_i = y_{i-1}; \quad y_i = x_{i-1} - y_{i-1} * (A \; div \; B)_i \; .$$

Первоначальные значения параметров следующие: *x=0,y=1.*

Применительно в алгоритму RSA уравнение принимает вид $e * d + \varphi(n) * y = 1$. В результате необходимо получить значение переменной d.

Пример

Найти d, зная что e=7,φ(n)=40.

Решение:

Представим решение в виде таблицы. В рассматриваемом примере $A = e=7$, $B(\varphi(n)=40$

A	B	A mod B	A div B	x	y
7	40	7	0		
40	7	5	5		
7	5	2	1		
5	2	1	2		
2	1	0	2		

Заполняем два последних столбца таблицы, начиная с последней строчки:

A	B	A mod B	A div B	x	y
7	40	7	0	-17	-17
40	7	5	5	3	-17
7	5	2	1	-2	3
5	2	1	2	1	-2
2	1	0	2	0	1

Таким образом получаем, что искомый параметр $d = -17$

Получим положительное значение параметра d:

$$d = y \bmod \varphi(n) = -17 \bmod 40 = 23$$

Шифрование

Отправитель разбивает свое сообщение на блоки m_i, длина которых в битах не превышает $\log_2 n$. Если блок имеет меньшую длину, то он добавляется нулями в двоичном представлении до необходимой величины.

Для каждого m_i вычисляется c_i, такое что $c_i = m_i^e \bmod n$

Дешифрование

Чтобы получить открытый текст необходимо каждый блок зашифрованного текста дешифровать по следующему правилу: $m_i = c_i^d \bmod n$

Рассмотрим конкретный пример:

Выбираем два простых числа p=7,q=17.

Вычисляем n=p·q=7·17=119

Вычисляем φ (n)= (p-1)·(q-1) = 96

Выбираем e, удовлетворяющее требованиям алгоритма: e=5

Определяем d так, чтобы d*e≡1 mod 96. d = 77

Результирующие ключи открытый KU = {5, 119} и закрытый KR = {77, 119}.

Например, требуется зашифровать сообщение M = 19

$19^5 = 66$ (*mod* 119); C = 66

Для дешифрования вычисляется 66^{77} (mod 119) = 19

Комбинирование симметричных и асимметричных алгоритмов

Комбинирование двух криптографических подходов позволяет использовать достоинства каждого из типов алгоритмов. В основном

идея сводится к тому, что сообщение шифруется симметричным алгоритмом, что позволяет выиграть в скорости, т.к. сообщение может быть сколь угодно большим, а ключ симметричного алгоритма шифруется асимметричным алгоритмом.

Содержание работы

1. Реализовать криптоалгоритм RSA.

Требования к реализации:

- простые числа p и q генерируются программой или задаются из файла, где они представлены в битовой форме.
- Длина генерируемых значение p и q должна задаваться пользователем;
- сгенерированные ключи записываются и хранятся в разных файлах.

2. Реализовать систему шифрования/дешифрования, в которой основным алгоритмом шифрования/дешифрования является AES. Ассиметричный алгоритм RSA используется для шифрования/дешифрования ключа.

Требования к реализации:

- ключ для симметричного алгоритма должен генерироваться случайным образом;
- зашифрованный текст должен сохраняться в первом файле, дешифрованный текст - во втором, а зашифрованный асимметричным алгоритмом ключ симметричного алгоритма – в третьем файле;
- Приложение расшифровывает зашифрованный ключ симметричного шифрования. После этого расшифровывает шифртекст.

- программа должна уметь работать с текстом произвольной длины.

Контрольные вопросы

1. В чём заключается алгоритм RSA?
2. В чем заключается криптографическая стойкость алгоритм RSA?
3. Дано e=3, φ (n)= 16. Найти d.
4. Для чего и почему используют комбинированные криптоалгоритмы?
5. В чём заключаются достоинства и недостатки асимметричных алгоритмов?
6. В чём заключаются достоинства и недостатки симметричных алгоритмов?

Лабораторная работа №7

АЛГОРИТМЫ ХЕШИРОВАНИЯ. ЭЛЕКТРОННАЯ ПОДПИСЬ

Цель работы: изучить алгоритм хеширования SHA-256. Ознакомиться со схемами получение электронной подписи и овладеть навыками создания и проверки подлинности электронной подписи.

Краткие теоретические сведения

Однонаправленные хеш-функции

Однонаправленная функция H(M) применяется к сообщению произвольной длины M и возвращает сообщение фиксированной длины h:

$$h= H(M),$$

где h имеет длины M.

Многие функции позволяют вычислять значение фиксированной длины по входным данным произвольной длины, но у однонаправленных хеш-функций есть дополнительные свойства, делающие их однонаправленными:

1) зная M, легко вычислить h. Зная H, трудно вычислить M, такое что H(M)=h;

2) зная M, трудно вычислить такое M' такое, что H(M)=H(M').

Смысл однонаправленных хеш-функций состоит в обеспечении для M уникального идентификатора («отпечатка пальца»).

В некоторых приложениях однонаправленности недостаточно, необходимо, чтобы выполнялось другое требование, называемое

устойчивостью к столкновению: должно быть трудно вычислить такое M' такое, что H(M)=H(M')

Хеш-функция, которая удовлетворяет первым двум свойствам, называется простой или слабой хеш-функцией. Выполнение последнего свойства делает хеш-функцию сильной. Оно защищает против класса атак, известных как атака "день рождения".

Атака «день рождения»

В криптографии под атакой «дней рождения» понимают метод взлома шифров или поиска коллизий хеш-функций на основе парадокса дней рождения.

Для заданной хеш-функции f целью атаки является поиск коллизии второго рода. Для этого вычисляются значения f для случайно выбранных блоков входных данных до тех пор, пока не будут найдены два блока, имеющие один и тот же хеш. Таким образом, если f имеет N различных равновероятных выходных значений, и N является достаточно большим, то из парадокса о днях рождения следует, что, в среднем, после перебора $1.25*\sqrt{N}$ различных входных значений, будет найдена искомая коллизия.

Следующий протокол, впервые описанный Гидеоном Ювалом, показывает, как, если требование устойчивости к столкновению не выполняется, пользователь А может использовать вскрытие методом дня рождения для обмана пользователя Б.

1) Пользователь А готовит две версии контракта: одну, выгодную для пользователя Б, и другую, приводящую его к банкротству.

2) Пользователь А вносит несколько незначительных изменений в каждый документ и вычисляет хеш-функции. (Этими

изменениями могут быть действия, подобные следующим: замена «пробела» комбинацией «пробел»–«забой»–«пробел», вставка одного-двух «пробелов» перед возвратом каретки и т.д. Делая или не делая по одному изменению в каждой из 32 строк, пользователь А может легко получить 232 различных документа).

3) Пользователь А сравнивает хеш-значения для каждого изменения в каждом из двух документов, разыскивая пару, для которой эти значения совпадают. Он восстанавливает два документа, дающих одинаковое хеш-значение.

4) Пользователь А получает подписанную пользователем Б выгодную для него версию контракта, используя протокол, которым он подписывает только хеш-значения.

5) Спустя некоторое время, пользователь А подменяет контракт, подписанный пользователем Б, другим, который он не подписывал. Теперь пользователь А может убедить арбитра в том, что пользователь Б подписал другой контракт

Вследствие этого длина хеш-кода должна быть достаточно большой. Длина, равная 64 битам, в настоящее время не считается безопасной. Предпочтительнее, чтобы длина составляла не менее 100 битов.

Для удлинения хеш-значений, выдаваемых конкретной хеш-функцией, был предложен следующий метод:

1) Для сообщения с помощью одной из однонаправленных хеш-функций генерируется хеш-значение.

2) Хеш-значение добавляется к сообщению.

3) Генерируется хеш-значение объединения сообщения и хеш-значения этапа 1.

4) Создаётся большее хеш-значение, состоящее из объединения хеш-значения этапа 1 и хеш-значения этапа 3.

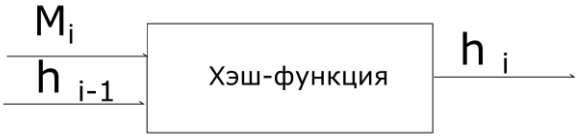
Этапы 1–4 повторяются нужное количество раз для обеспечения требуемой длины хеш-значения.

Обзор однонаправленных хеш-функций

Нелегко построить функцию, вход которой имеет произвольный размер, а тем более сделать её однонаправленной. В реальном мире однонаправленные хеш-функции строятся на идее функции сжатия. Такая однонаправленная функция выдаёт хеш-значение длины при заданных входных данных большей длины. Входами функции сжатия являются блок сообщения и выход предыдущего блока текста. Выход представляет собой хеш-значение всех блоков, т.е. хеш-значение блока Mi равно

$$h_i = f(M_i, h_{i-1})$$

Это хеш-значение вместе со следующим блоком сообщения становится следующим входом функции сжатия. Хеш-значением всего сообщения будет хеш-значение последнего блока.



Хешируемый вход должен каким-то способом содержать бинарное представление длины всего сообщения. Таким образом, преодолевается потенциальная проблема, вызванная тем, что

сообщения различной длины могут давать одно и то же хеш-значение. Иногда такой метод называется MD-усилением.

В качестве однонаправленных хеш-функций можно использовать симметричные блочные алгоритмы шифрования. Самый очевидный способ – это шифрование сообщения в режиме CBC и CFB с помощью фиксированного ключа и ключа инициализации, хеш-значением будет последний блок шифр текста.

Более хороший способ использует в качестве ключа блок сообщения, в качестве входа – предыдущее хеш-значение, а выходом служит текущее хеш-значение.

Действительные хеш-функции ещё сложнее. Размер блока обычно совпадает с длиной ключа, размером хеш-значения будет длина блока. Т.к. большинство блочных алгоритмов 64-битные, то спроектирован ряд схем, дающих хеш-значение, в два раза большее длины блока.

При условии, что хеш-функция правильна, безопасность этой схемы основана на безопасности используемой блочной функции. Однако есть и исключения. Дифференциальный криптоанализ лучше работает против блочных функций в хеш-значениях, чем против блочных функций, используемых для шифрования: ключ известен, поэтому можно использовать различные приёмы. Для успеха нужна только одна правильная пара, и можно генерировать столько выбранного открытого текста, сколько нужно. Полезной мерой для хеш-функций, основанных на блочных шифрах, является скорость хеширования или количество n-битовых блоков сообщения, обрабатываемых при шифровании. Чем выше скорость хеширования, тем быстрее алгоритм

Хеш-функция SHA-256

SHA-256 представляет собой криптографическую функцию хеширования. Данная хеш-функция была принята в качестве стандарта в 2001 году.

Основные показатели (измеряются в битах):

Длина сообщения: менее 2^{64}

Длина блока: 512

Длина слова: 32

Количество итераций в цикле: 64

Длина хеш-значения: 256

Алгоритм работает с данными, разбитыми на куски в 512 бит. Раунд в алгоритме SHA-256 повторяется 64 раза. Все операции выполняются над 32-битными словами. Результатом каждой функции также является 32-битное слово.

SHA-256 использует шесть логических функций: конъюнкция, побитовое "И", исключающее "ИЛИ", логический сдвиг вправо на n бит (SHR^n), циклический сдвиг вправо на n бит ($ROTR^n$), сложение по модулю 2^{32}:

$$Ch(x, y, z) = (x \wedge y) \oplus (\neg x \wedge z)$$

$$Maj(x, y, z) = (x \wedge y) \oplus (x \wedge z) \oplus (y \wedge z)$$

$$\sum_0^{256}(x) = ROTP^2(x) \oplus ROTR^{13}(x) \oplus ROTR^{22}(x)$$

$$\sum_1^{256}(x) = ROTP^6(x) \oplus ROTR^{11}(x) \oplus ROTR^{25}(x)$$

$$\sigma_0^{256}(x) = ROTP^7(x) \oplus ROTR^{18}(x) \oplus SHR^3(x)$$

$$\sigma_1^{256}(x) = ROTP^{17}(x) \oplus ROTR^{19}(x) \oplus SHR^{10}(x)$$

Добавление недостающих блоков

Сообщение добавляется таким образом, чтобы его длинна была кратна 448 по модулю 512, то есть $dlina = 448 \bmod 512$. Добавление происходит всегда и включает в себя 1 и необходимое количество нулей. После этого сообщение представляет собой последовательность из N блоков по 512 бит .

Основной алгоритм

Определяются 8 32-битных переменных: a,b,c,d,e,f,g,h. Данные переменные представляют собой промежуточные значения хеш-кода.

Над каждым полученным блоком исходного сообщения производиться 64 циклические обработки:

$$T_1 = h + \Sigma_1^{256}(e) + Ch(e, f, g) + K_t^{256} + W_t$$
$$T_2 = \Sigma_0^{256}(a) + Maj(a, b, c)$$
$$h = g$$
$$g = f$$
$$f = e$$
$$e = d + T_1$$
$$d = c$$
$$c = b$$
$$b = a$$
$$a = T_1 + T_2$$

Ki - шестьдесят четыре 32-битных константы, каждая из которых является первыми 32-мя битами дробной части кубических корней первых 64 простых чисел.

Wt вычисляются из очередного блока сообщения по следующим правилам:

$$\begin{cases} W_t = M_t^i, 0 \le t \le 15 \\ \sigma_1^{256}(W_{t-2}) + W_{t-7} + \sigma_{01}^{256}(W_{t-15}) + W_{t-16}, 16 \le t \le 63 \end{cases}$$

Каждое i-ое промежуточное значение хеша вычисляется так:

$H_0^i = a + H_0^{i-1}$

$H_1^i = b + H_1^{i-1}$

$H_2^i = c + H_2^{i-1}$

$H_3^i = d + H_3^{i-1}$

$H_4^{i-} = e + H_0^{4-1}$

$H_5^i = f + H_5^{i-1}$

$H_6^i = g + H_6^{i-1}$

$H_7^i = h + H_7^{i-1}$

Электронная подпись

Использование симметричных систем не позволяет полностью решить проблему проверки подлинности получаемого сообщения. Факт наличия секретного ключа у отправиля и получателя дает второму участнику возможность формирования нового сообщения с использованием имеющегося аутентификатора.

Для исключения подобных ситуаций необходимо присутствие третьего участника (доверенного лица), которому должна посылаться копия сообщения. Однако, такой сценарий является не самым удобным. Решением проблемы стало использование ассиметричных алгоритмов шифрования.

В соответствии с Федеральным законом "Об электронной подписи" от 06.04.2011 под электронной подписью понимается информация в электронной форме, которая присоединена к другой информации в электронной форме (подписываемой информации) или иным образом

связана с такой информацией и которая используется для определения лица, подписывающего информацию.

Проверка подлинности сообщения осуществляется с помощью открытого ключа автора. Эти данные доступны любому пользователю.

К электронной подписи предъявляются следующие требования:

1) уникальность (т. е. возможность использования ее только одним определенным пользователем);

2) невозможность отказа от выполненной подписи;

3) допустимость верификации подписи любым пользователем;

4) простота выполнения процедур создания и верификации подписи;

5) возможность выполнения для больших объемов информации (файлов, дисков и т. д.).

В настоящее время применяется несколько алгоритмов электронной подписи:

1) RSA (наиболее популярен);

2) DigitalSignatureAlgorithm, DSA (одно из средств электронной подписи, применяемое в стандарте DSS);

3) алгоритм Эль-Гамаля (иногда можно встретить);

4) алгоритм, который применяют в стандарте ГОСТ Р34.10-94 (в основе лежит DSA и является вариацией подписи Эль-Гамаля);

5) так же существуют алгоритмы подписей, в основе которых лежит криптография эллиптических кривых; они похожи на все прочие, но в некоторых ситуациях работают эффективнее.

Электронная подпись на основе алгоритма RSA

Схема использования алгоритма RSA при большом модуле N практически не позволяет злоумышленнику получить

закрытый ключ и прочитать зашифрованное сообщение. Однако она дает возможность злоумышленнику подменить сообщение от пользователя А к пользователю Б, так как пользователь А шифрует свое сообщение открытым ключом, полученным от пользователя Б по открытому каналу связи. Так как открытый ключ передается по открытому каналу, любой может получить его и использовать для подмены сообщения. Избежать этого можно, используя более сложные протоколы, например, следующий.

Пусть, как и раньше, пользователь А хочет передать пользователю Б сообщение, состоящее из нескольких блоков m_i. Перед началом сеанса связи абоненты генерируют следующие открытые и закрытые ключи.

	Открытый ключ	Закрытый ключ
Пользователь А	N_A, d_A	e_A
Пользователь Б	$N_Б$, $d_Б$	$e_Б$

В результате каждый пользователь имеет свои собственные открытый (состоящий из двух частей) и закрытый ключи. Затем пользователи обмениваются открытыми ключами. Это подготовительный этап протокола.

Основная часть протокола состоит из следующих шагов.

Сначала пользователь А вычисляет числа $c_i = m_i^{e_A} \bmod N_A$, то есть шифрует сообщение своим закрытым ключом. В результате этих действий пользователь А подписывает сообщение.

Затем пользователь А вычисляет числа $g_i = c_i^{d_B} \bmod N_B$, то есть шифрует то, что получилось на шаге 1 открытым ключом

пользователя Б. На этом этапе сообщение шифруется, чтобы никто посторонний не мог его прочитать.

Последовательность чисел g_i передается к пользователю Б.

Пользователь Б получает g_i и вначале вычисляет последовательно числа $c_i = g_i^{e_B} \bmod N_B$, используя свой закрытый ключ. При этом сообщение расшифровывается.

Затем пользователь Б определяет числа $g_i = c_i^{d_A} \bmod N_A$, используя открытый ключ пользователя А. За счет выполнения этого этапа производится проверка подписи пользователя А.

В результате пользователь Б получает исходное сообщение и убеждается в том, что его отправил именно пользователь А. Данная схема позволяет защититься от нескольких видов возможных нарушений, а именно:

- пользователь А не может отказаться от своего сообщения, если он признает, что секретный ключ известен только ему;
- нарушитель без знания секретного ключа не может ни сформировать, ни сделать осмысленное изменение сообщения, передаваемого по линии связи.

Данная схема позволяет избежать многих конфликтных ситуаций. Иногда нет необходимости зашифровывать передаваемое сообщение, но нужно его скрепить электронной подписью. В этом случае из приведенного выше протокола исключаются шаги 2 и 4, то есть текст шифруется закрытым ключом отправителя, и полученная последовательность присоединяется к документу. Получатель с помощью открытого ключа отправителя расшифровывает прикрепленную подпись, которая, по сути, является зашифрованным

повторением основного сообщения. Если расшифрованная подпись совпадает с основным текстом, значит, подпись верна.

Существуют и другие варианты применения алгоритма RSA для формирования ЭП. Например, можно шифровать (то есть подписывать) открытым ключом не само сообщение, а хеш-код от него.

Для осуществления подписи сообщения $M=M_1M_2M_3….M_n$ необходимо вычислить хеш- функцию $m=h(M=M_1M_2M_3….M_n)$, которая ставит в соответствие сообщению M число m. На следующем шаге достаточно снабдить подписью только число m, и эта подпись будет относиться ко всему сообщению M.

Далее по алгоритму RSA вычисляются ключи (e, n) и (d, n).

Затем вычисляется $s = m^d \, mod \, n$

Число s– это и есть электронная подпись. Она просто добавляется к сообщению и получается подписанное сообщение (M, s)

Теперь каждый, кто знает параметры подписавшего сообщение (т.е. числа e и N), может проверить подлинность подписи.

Для этого необходимо проверить выполнение равенства $h(M) = s^e mod \, n.$

Возможность применения алгоритма RSA для получения электронной подписи связана с тем, что секретный и открытый ключи в этой системе равноправны. Каждый из ключей, d или e, могут использоваться как для шифрования, так и для расшифрования. Это свойство выполняется не во всех криптосистемах с открытым ключом.

Алгоритм RSA можно использовать также и для обмена ключами.

Электронная подпись на основе алгоритма Эль-Гамаля

Принцип создания и проверки подписи

Алгоритм Эль-Гамаля также можно использовать для формирования электронной подписи. Группа пользователей выбирает общие параметры P и A. Затем каждый абонент группы выбирает свое секретное число Xi, такое что 1<Xi<P-1, и вычисляет соответствующее ему открытое число Y_i : $Y_i = A^{X_i} \bmod P$. Таким образом, каждый пользователь получает пару (закрытый ключ; открытый ключ) = (Xi, Yi). Открытые ключи пользователей могут храниться в общей базе системы распределения ключей и при необходимости предоставляться всем абонентам системы.

Сообщение, предназначенное для электронной подписи, должно быть представлено в виде числа, меньшего модуля P. При большом размере сообщение разбивается на блоки необходимого размера. В некоторых случаях подписывается не само сообщение, а значение хеш-функции от него. В любом варианте электронная подпись вычисляется в зависимости от некоторого числа m (m < P).

Пусть пользователь A хочет подписать свое сообщение цифровой подписью и передать его пользователю Б. В этом случае алгоритм действий следующий.

Пользователь A выбирает случайное секретное число k, взаимно простое с P-1, и вычисляет число a : $a = A^{k_i} \bmod P$

Затем с помощью расширенного алгоритма Евклида необходимо найти значение b в следующем уравнении:

$$m = (x \cdot a + k \cdot b) \bmod (P-1)$$

Пара чисел (a, b) будет электронной подписью сообщения m.

Сообщение m вместе с подписью (a, b) отправляется пользователю Б.

Пользователь Б получает сообщение m и с использованием открытого ключа пользователя А и вычисляет два числа по следующим формулам: $c_1 = Y_1 * a^b \mod P$

$c_2 = A^m \mod P.$

Если c1 =c2, то электронная подпись пользователя А верная. Для подписывания каждого нового сообщения должно каждый раз выбираться новое значение k.

Подписи, созданные с использованием алгоритма Эль-Гамаля, называются рандомизированными, так как для одного и того же сообщения с использованием одного и того же закрытого ключа каждый раз будут создаваться разные подписи (a, b), поскольку каждый раз будет использоваться новое значение k. Подписи, созданные с применением алгоритма RSA, называются детерминированными, так как для одного и того же сообщения с использованием одного и того же закрытого ключа каждый раз будет создаваться одна и та же подпись.

Пример вычисления и проверки электронной подписи

Пусть абоненты, обменивающиеся через Интернет зашифрованными сообщениями, имеют следующие общие параметры: P = 11, A = 7.

Один из пользователей этой системы связи хочет подписать свое сообщение m=5 электронной подписью, сформированной по алгоритму Эль-Гамаля. Вначале он должен выбрать себе закрытый ключ, например, X=3 и сформировать открытый ключ $Y= 7^3 \mod 11 = 2$. Открытый ключ может быть передан всем

заинтересованным абонентам или помещен в базу данных открытых ключей системы связи.

Затем пользователь выбирает случайное секретное число k, взаимно простое с P-1. Пусть k=9 (9 не имеет общих делителей с 10). Далее необходимо вычислить число $a=7^9 \bmod 11=8$.

После этого с помощью расширенного алгоритма Евклида находится значение b в уравнении:

$$m = (x\cdot a + k \cdot b) \bmod (P-1)=(3\cdot 8+9\cdot b) \bmod 10$$

Решением последнего уравнения будет значение b=9.

Таким образом, пара чисел (8, 9) будет электронной подписью сообщения m=5.

Если любой другой пользователь сети желает проверить электронную подпись в сообщении, он должен получить из базы данных открытый ключ первого пользователя (он равен 2), вычислить два числа c1 и c2 и сравнить их.

$c_1=2^8\ 8^9 \bmod 11=10$

$c_2= 7^5 \bmod 11=10$

Так как c1 =c2=10, то электронная подпись первого пользователя верная.

Содержание работы

1. Разработать программу, выдающую дайджест сообщения произвольной длины, на основе алгоритма SHA- 256.
2. Вручную получить и проверить ЭЦП на основе алгоритма Эль-Гамаля.
3. Реализовать приложение, позволяющее вычислять ЭЦП, сформированную по алгоритмам RSA и Эль-Гамаля. Программа

должна работать с простыми числами большими 2^{64}. С помощью разработанного приложения проверить решение пункта 2.

Варианты заданий.

№	ЭЦП по алгоритму Эль-Гамаля
	Секретные параметры
1	X=11,k=3
2	X=10,k=15
3	X=3,k=13
4	X=6,k=13
5	X=13,k=19
6	X=17,k=5
7	X=8,k=17
8	X=15,k=15
9	X=14,k=17
10	X=13,k=19

Контрольные вопросы

1. Что такое хеш-функция, для чего она используется? В чём заключается устойчивость к столкновениям?
2. Как обмануть подписчика, если требование устойчивости к столкновению не выполняется?
3. Что такое коллизия хеш-функций?
4. Создание хеш-функции алгоритмом SHA-256?
5. Для чего нужна цифровая подпись? Основные свойства цифровой подписи.
6. Какие схемы цифровой подписи существуют? Какая схема самая распространенная и почему?

7. Как осуществляется подпись RSA? В чем отличие подписи RSA от алгоритма шифрования RSA?

8. Как осуществляются подпись и проверка на подлинность подписи по алгоритму Эль-Гамаля?

Лабораторная работа №8

ЭЛЕКТРОННАЯ ПОДПИСЬ НА ОСНОВЕ ЭЛЛИПТИЧЕСКОЙ КРИВОЙ

Цель работы: изучить алгоритм получения ЭЦП на основе эллиптической кривой.

Краткие теоретические сведения

Теория эллиптических кривых развивается во многих направлениях. Отдельного внимания заслуживает рассмотрение вопроса использования алгоритмов на основе эллиптической кривой в криптографии.

Преимущества использования **ESDSA** (Elliptic Curve Digital Signature Algorithm):

1) Степень защищенности на каждый бит выше, чем при использовании в системах с открытым ключом;

2) Быстродействие при программной и аппаратной реализации;

3) Успешно встраиваются в работу стандартных схем ассиметричного шифрования (шифрование Эль-Гамаля, RSA)

Эллиптическая кривая и эллиптическая группа

Эллиптической кривой называют множество пар точек (X,Y), удовлетворяющих уравнению: $y^2 = ax^3 + bx + c$.

Эллиптическая группа $E_p(a,b)$ представляет собой набор точек (x,y) с целыми положительными координатами, $x < p$ и $y < p$, которые удовлетворяют соотношению: $y^2 = x^3 + ax + b \pmod p$.

Алгоритм формирования элементов эллиптической группы:

1) Для всех значений x (0 <x <p) вычисляется значение $x^3 + ax + b \pmod p$.

2) Для каждого значения из шага 1 определяется квадратный корень по модулю M и элемент включается в группу $E_p(a,b)$, если результат положительный.

Использование эллиптических кривых для создания стандарта цифровой подписи

В качестве секретного ключа выбирается некоторое случайное число x. Открытым ключом являются координаты точки на эллиптической кривой P, определяемую как P = x*Q, где Q - специальным образом выбранная точка эллиптической кривой («базовая точка»). Координаты точки Q вместе с коэффициентами уравнения, задающего кривую, являются параметрами схемы подписи и должны быть известны всем участникам обмена сообщениями.

Выбор точки Q зависит от используемых алгоритмов и весьма непрост. Так, стандарт ГОСТ 34.10-2001 определяет, что точка Q должна иметь порядок q, где q — простое число с «хорошими алгебраическими свойствами». Число q довольно велико (2254 < q< 2256).

Алгоритм ЭП ESDSA

Алгоритм состоит из трех основных этапов: генерации, подписывания и проверки.

Алгоритм генерации

1) Выбирается эллиптическая кривая E, определённая на Z_p. Количество точек в $E(Z_p)$ должно быть сравнимо с большим n.

2) Выбирается точка P є $E(Z_p)$ порядка n.

3) Выбирается случайное число d, которое лежит в интервале от 1 до n включительно.

4) Вычисляем Q є d·P (точка P, сложенная сама с собой d-1 раз)

5) Секретный ключ – d.

6) Открытый ключ (E,P,n,Q).

Этап формирования подписи

1) Вычисляем случайное число k, которое удовлетворяет условию $1 \le k \le n-1$.

2) Вычисляется k*P= (x_1,y_1)

3) r = x_1 mod n

4) Если r ≠0 переходим к шагу 3, иначе к шагу 1

5) Вычисляем k^{-1}mod n (используется расширенный алгоритм Евклида)

6) Вычисляем $s = k^{-1} (h(M)+d*r) \bmod n$

7) Если s ≠0 переходим к шагу 5, иначе шаг 1

8) Подписью сообщения M является пара чисел (r,s)

Проверка подписи

1) Если r,s – целые числа, то переходим к шагу 2, иначе подпись не корректна (причем r,s є [1..n-1]).

2) Вычисляем $w=s^{-1} \bmod n$ и h(M).

3) Вычисляем $u_1=h(M')w \bmod n$ и $u_2=r*w \bmod n$.

4) Вычисляется $u_1*P+u_2*Q=(x_0,y_0)$.

5) Вычисляется $v = x_0 \bmod n$.

6) Подпись верна если v=r.

Задание к лабораторной работе

1. Разработать приложение, реализующее ESDSA. В качестве используемой хеш-функции выбирается SHA-256. Примеры для расчета брать из стандарта NIST (http://csrc.nist.gov/groups/ST/toolkit/documents/dss/NISTReCur.pdf)

Контрольные вопросы

1. Что такое эллиптическая кривая?
2. Дать определение "нулевой точки".
3. Сформулировать "проблему дискретного логарифма эллиптической кривой"
4. Что такое эллиптическая группа.
5. Дано число p=23. Определить параметры a,b и построить эллиптическую группу E_p (a,b).

Расчетно-графическое задание.

Написать реферат по предложенным темам.

Вариант	Тема
1	Безопасность в Интеренете.
2	Источники угроз безопасности персональных данных.
3	Отличительные особенности информационной безопасности РФ.
4	Методы защиты информации в современных ОС.
5	Роль информационной безопасности в современном мире.
6	Проблемы обеспечения информационной безопасности.
7	Системы обнаружения вторжений.
8	Обеспечение защиты данных в беспроводных сетях.
9	Проблемы информатизации общества. Способы защиты своей индивидуальности.
10	Нововведения в законодательную базу РФ в области информационной безопасности.
11	Методы оценки рисков безопасности.
12	Особенности защиты информации в сетях различной архитектуры.
13	Компьютерное пиратство. Методы борьбы.
14	Защита данных с помощью биометрики.
15	Анализ возможных каналов утечки информации.

Библиографический список

1. Криптографические методы защиты информации: методические указания к выполнению лабораторных работ и индивидуальных домашних заданий для студентов специальности 090303,65-Информационная безопасность автоматизированных систем/ сост.: Е.Н. Сергиенко, Д.О. Давыденко, И.М. Идеменко, А.А. Хохлов. - Белгород: Изд-во БГТУ, 2014-82 с.

2. https://www.intuit.ru/studies/courses/552/408/info курс "Математика криптографии и теория шифрования"

3. Криптографические методы защиты информации: учебное пособие для студ. вузов/ С.Б. Гашков, Э.А. Применко, М.А. Черепнев - М.: Изд. центр "Академия", 2010-304с.

4. Курс "Криптографические методы защиты информации". https://www.intuit.ru/studies/courses/13837/1234/info

5. Алгоритм блочного симметричного шифрования Advanced Encryption Standard (AES) Технический отчет CELLAES-01 К.С. Пан, М.Л. Цымблер http://crypto.pp.ua/wp-content/uploads/2010/03/aes.pdf

6. Криптография и безопасность сетей: Учебное пособие/Фороузан Б.А.; перевод с англ. под ред. А.Н.Берлина.- М. : Интернет-Университет Информационных Технологий: БИНОМ. Лаборатория знаний,2010-784с.

7. Математика криптографии: методические указания к выполнению лабораторных работ и индивидуальных домашних заданий для студентов специальности 090303,65-Информационная безопасность / сост.: Е.Н. Сергиенко, С.А. Панин, И.А. Пригорнев, А.С. Чурилов. - Белгород: Изд-во БГТУ, 2014-59 с.

8. Защита информации. Учебное пособие для бакалавриата и магистратуры. 2-е издание -Юрайт, 2016-261 с.

Учебное издание

ИНФОРМАЦИОННАЯ БЕЗОПАСНОСТЬ

Методические указания к выполнению
лабораторных работ и РГЗ для студентов очной формы обучения
направлений бакалавриата
09.03.02 - Информационные системы и технологии и
09.03.03 - Прикладная информатика

Составители: **Жданова** Светлана Ивановна
Рога Сергей Николаевич

Подписано в печать 2.02.18. Формат 60x84 /16. Усл. печ. л. 5,1. Уч.-изд.л. 5,6.
Тираж 90 экз. Заказ Цена
Отпечатано в Белгородском государственном технологическом университете
им. В. Г. Шухова
308012, г. Белгород, ул. Костюкова, 46