МИНИСТЕРСТВО ЦИФРОВОГО РАЗВИТИЯ, СВЯЗИ
И МАССОВЫХ КОММУНИКАЦИЙ РОССИЙСКОЙ ФЕДЕРАЦИИ
Северо-Кавказский филиал
ордена Трудового Красного Знамени федерального государственного
бюджетного образовательного учреждения высшего образования
«Московский технический университет связи и информатики»

Методические указания
по проведению практических занятий
по дисциплине

КРИПТОГРАФИЧЕСКИЕ ПРОТОКОЛЫ

Направление подготовки 11.03.02
Инфокоммуникационные технологии и системы связи
профиль Защищенные инфокоммуникационные системы

Ростов-на-Дону
2022

Методические указания
по проведению практических занятий
по дисциплине

КРИПТОГРАФИЧЕСКИЕ ПРОТОКОЛЫ

Составитель И.А. Енгибарян доцент кафедры ИТСС

Рассмотрено и одобрено
на заседании кафедры «ИТСС»
Протокол от «19» декабря 2022 г., №5

ПРАКТИЧЕСКОЕ ЗАНЯТИЕ №1

Криптоанализ и дешифрование. Правило Керкхоффса. Криптография и теория чисел. Модулярная арифметика

адание 1.1. Найти наибольший общий делитель НОД(a,b) двух чисел $a \in Z$, $b \in N$ методом Евклида.

Алгоритм и расчетные формулы метода Евклида:

$$r_{j-2} = q_j r_{j-1} + r_j,\ q_j = \left\lfloor \frac{r_{j-2}}{r_{j-1}} \right\rfloor,\ j = 0,1,2\ldots, \text{ где } r_{-2} = a,\ r_{-1} = b.$$

Решением задачи является последний ненулевой остаток $r_j > 0$.

Пример. Найти НОД$(1547,560)$. Решение приведено в табл. 1.1.

Таблица 1.1 – Выполнение алгоритма Евклида для примера

шаг	кратность	разложение	остаток
$j=1$	$q_1 = \left\lfloor \frac{a}{b} \right\rfloor = \left\lfloor \frac{1547}{560} \right\rfloor = 2$	$a = q_1 b + r_0$: $1547 = 2 \cdot 560 + 427$	$r_0 = 427$
$j=2$	$q_2 = \left\lfloor \frac{b}{r_0} \right\rfloor = \left\lfloor \frac{560}{427} \right\rfloor = 1$	$b = q_2 r_0 + r_1$: $560 = 1 \cdot 427 + 133$	$r_1 = 133$
$j=3$	$q_3 = \left\lfloor \frac{r_0}{r_1} \right\rfloor = \left\lfloor \frac{427}{133} \right\rfloor = 3$	$r_0 = q_3 r_1 + r_2$: $427 = 3 \cdot 133 + 28$	$r_2 = 28$
$j=4$	$q_4 = \left\lfloor \frac{r_1}{r_2} \right\rfloor = \left\lfloor \frac{133}{28} \right\rfloor = 4$	$r_1 = q_4 r_2 + r_3$: $133 = 4 \cdot 28 + 21$	$r_3 = 21$
$j=5$	$q_5 = \left\lfloor \frac{r_2}{r_3} \right\rfloor = \left\lfloor \frac{28}{21} \right\rfloor = 1$	$r_2 = q_5 r_3 + r_4$: $28 = 1 \cdot 21 + 7$	$r_4 = 7$
$j=6$	$q_6 = \left\lfloor \frac{r_3}{r_4} \right\rfloor = \left\lfloor \frac{21}{7} \right\rfloor = 3$	$r_3 = q_6 r_4 + r_5$: $21 = 3 \cdot 7 + 0$	$r_5 = 0$

Получено: $r_5 = 0$, следовательно, выполнение алгоритма окончено. Решением является предшествующий (ненулевой) остаток $r_4 = 7$: НОД$(1547,560) = 7$.

В качестве индивидуальных заданий для демонстрации приобретенных навыков по данному заданию может выступать определение НОД для ключей в лабораторной работе по освоению криптографической системы RSA (табл. 2.2).

Задание 1.2. Найти результат преобразования методом, основанным на теореме Эйлера.

Функция Эйлера: $\varphi(n)=\left|\left\{0\le b<n \mid НОД(b,n)=1\right\}\right|$ обладает следующими ключевыми свойствами:

1) $\varphi(1)=1$;

2) для любого простого p: $\varphi(p)=p-1$;

3) для любого простого p в степени α: $\varphi\left(p^{\alpha}\right)=p^{\alpha}-p^{\alpha-1}$;

4) $\forall m,n\in N \mid НОД(m,n)=1: \varphi(m\cdot n)=\varphi(m)\cdot\varphi(n)$.

Используется теорема Эйлера:

$$\forall a,m\in N \mid НОД(a,m)=1: a^{\varphi(m)}\equiv 1(\mathrm{mod}\, m),$$

и следствие из нее: если НОД$(a,m)=1$ и n' – наименьший неотрицательный вычет n по модулю $\varphi(m)$, то $a^n\equiv a^{n'}(\mathrm{mod}\, m)$.

Пример. Провести преобразование: $5^{17}\bmod 7$

Решение. $a=5; n=17; m=7$. При этом 7 – простое число.

Находим $\varphi(m)=\varphi(7)=7-1=6$;

$n'=17\bmod(6)=(2\cdot 6+5)\bmod(6)=5$;

$5^{17}\bmod 7=5^5\bmod 7=3125\bmod 7=(446\cdot 7+3)\bmod 7=3$.

Примеры индивидуальных заданий для проверки умений по заданию 1.2 приведены в табл. 1.4.

Таблица 1.4 – Примеры задач для проверки умений

№	задание	№	задание	№	задание
1	$17^{31}\bmod 7$	7	$17^{121}\bmod 7$	13	$3^{213}\bmod 9$
2	$3^{311}\bmod 7$	8	$3^{121}\bmod 7$	14	$17^{21}\bmod 9$
3	$17^{213}\bmod 7$	9	$2^{213}\bmod 5$	15	$5^{63}\bmod 7$
4	$2^{61}\bmod 5$	10	$3^{255}\bmod 7$	16	$5^{21}\bmod 7$
5	$2^{111}\bmod 5$	11	$5^{213}\bmod 7$	17	$5^{213}\bmod 7$
6	$3^{61}\bmod 7$	12	$2^{63}\bmod 7$	18	$7^{17}\bmod 5$

Задание 1.3. Найти обратное значение числа по модулю:

– при помощи определения кратности k («по определению»);

– с использованием функции Эйлера.

Сравнить полученные результаты и оценить трудоемкость.

По определению обратные числа по модулю означают, что $a\cdot b\equiv 1(\mathrm{mod}\, m)$. Обратное число $b\equiv a^{-1}(\mathrm{mod}\, m)$ гарантированно

существует, если $\text{НОД}(a,m)=1$ (a и m – взаимно простые). При этом выполнено диофантово уравнение: $a\cdot b-k\cdot m=1$, где k – кратность m в числе $a\cdot b$. Внимание: число k должно быть получено *целым* числом.

Пример: найти $x=9^{-1}(\text{mod}\,5)$.

Решение «по определению». Получаем уравнение $9\cdot x-k\cdot 5=1$;

$$x=(1+k\cdot 5):9.$$

Производим последовательный перебор возможных кратностей k как чисел последовательности натуральных чисел $\{1,2,3,4,....\}$ до тех пор, пока не получим x как *целое* число. В данном примере при $k=7$ получено целое число $x=4$.

Итак, получено $9^{-1}(\text{mod}\,5)=4$.

Проверка: $9\cdot 4-7\cdot 5=36-35=1$.

Решение по теореме Эйлера: $a^{\varphi(m)}\equiv 1(\text{mod}\,m)$. Домножим левую и правую части этого уравнения на $a^{-1}(\text{mod}\,m)$. Получаем правило нахождения числа, обратного по модулю (в кольце), *с использованием функции Эйлера:*

$$a^{\varphi(m)-1}\equiv a^{-1}(\text{mod}\,m).$$

Пример тот же: $x=9^{-1}(\text{mod}\,5)$.

Решение: $a=9, m=5$. $\text{НОД}(9,5)=1$, $m=5$ – простое число.

Находим $\varphi(m)=\varphi(5)=5-1=4$;

$x=9^{4-1}(\text{mod}\,5)=729(\text{mod}\,5)=(145\cdot 5+4)(\text{mod}\,5)=4$, т.е. $b=x=4$.

Получен тот же результат без перебора неподходящих вариантов k. Если в дальнейшем потребуется k как коэффициент диофантова уравнения, его можно будет определить из уравнения $a\cdot b-k\cdot m=1$.

Примеры индивидуальных заданий для проверки умений по заданию 1.3 приведены в табл. 1.5.

Таблица 1.5 – Примеры задач для проверки умений

№	задание	№	задание	№	задание
1	$7^{-1}(\text{mod}\,9)$	7	$7^{-1}(\text{mod}\,17)$	13	$9^{-1}(\text{mod}\,23)$
2	$19^{-1}(\text{mod}\,5)$	8	$5^{-1}(\text{mod}\,9)$	14	$7^{-1}(\text{mod}\,23)$
3	$17^{-1}(\text{mod}\,5)$	9	$19^{-1}(\text{mod}\,7)$	15	$7^{-1}(\text{mod}\,13)$

№	задание	№	задание	№	задание
4	$5^{-1}(\mathrm{mod}\,27)$	10	$17^{-1}(\mathrm{mod}\,9)$	16	$7^{-1}(\mathrm{mod}\,5)$
5	$5^{-1}(\mathrm{mod}14)$	11	$17^{-1}(\mathrm{mod}\,7)$	17	$5^{-1}(\mathrm{mod}\,7)$
6	$79^{-1}(\mathrm{mod}17)$	12	$17^{-1}(\mathrm{mod}\,79)$	18	$7^{-1}(\mathrm{mod}11)$

Задание 1.4. Найти результат преобразования методом цепочек.

При больших значениях чисел $a^{n'}$(задание 1.2) или $a^{\varphi(m)-1}$ (задание 1.3) определение вычетов по $(\mathrm{mod}\,m)$ может быть проведено с использованием метода цепочек модулярной арифметики:

$$(a \pm b)\bmod m = \big((a \bmod m) \pm (b \bmod m)\big)\bmod m;$$

$$(a * b)\bmod m = \big((a \bmod m) * (b \bmod m)\big)\bmod m;$$

$$\big(a*(b\pm c)\big)\bmod m = \big(\big((a*b)\bmod m\big) \pm \big((a*c)\bmod m\big)\big)\bmod m.$$

Пример 1. Пусть требуется представить в форме цепочки $a^8(\mathrm{mod}\,m)$. Степень четная, более того $8 = 2^3$.

Цепочка преобразований

$$(((a^2(\mathrm{mod}\,m)^2 \bmod m)^2 \bmod m.$$

При использовании сложных чисел в выражении $a^n(\mathrm{mod}\,m)$ как в качестве a, так и в качестве степени n (n' или $(\varphi(m)-1)$) может быть использована факторизация чисел (разложение их на простые сомножители).

Пример 2. Найти методом цепочек $a^n(\mathrm{mod}\,m) = 15^{63}(\mathrm{mod}\,7)$.

Решение. $a = 15 = 3\cdot 5;\ n = 63 = 3^2\cdot 7.$

$$15^{63}(\mathrm{mod}\,7) = ((3^{63} \bmod 7)\cdot(5^{63} \bmod 7))\bmod 7 = (A\cdot B)\bmod 7;$$

$$A = 3^{63} \bmod 7 = 3^{3\cdot 3\cdot 7} \bmod 7 = ((3^3 \bmod 7)^3 \bmod 7)^7 \bmod 7;$$

$$3^3 \bmod 7 = 27 \bmod 7 = (3\cdot 7+6)\bmod 7 = 6;$$

$$6^3 \bmod 7 = 216 \bmod 7 = (30\cdot 7+6)\bmod 7 = 6;$$

$$6^7 \bmod 7 = 6^{2\cdot 2+3} \bmod 7 = ((6^{2\cdot 2} \bmod 7)\cdot(6^3 \bmod 7))\bmod 7;$$

$$6^{2\cdot 2} \bmod 7 = (6^2 \bmod 7)^2 \bmod 7;\ 6^2 \bmod 7 = 36 \bmod 7 = (5\cdot 7+1)\bmod 7 = 1;$$

$$1^2 \bmod 7 = 1;\ 6^7 \bmod 7 = (1\cdot 6)\bmod 7 = 6;\quad A = 6.$$

Аналогичными действиями для $5^{63} \bmod 7$ получаем $B = 6.$

$$15^{63}(\mathrm{mod}\,7) = A\cdot B \bmod 7 = (6\cdot 6)\bmod 7 = 36 \bmod 7 = (5\cdot 7+1)\bmod 7 = 1.$$

В прикладной криптографии (при использовании двоичного кода представления целых чисел) особое значение имеет возможность построения на основе метода цепочек алгоритма повторного возведения в квадрат, который реализуется при выполнении заданий лабораторного практикума.

Обозначим промежуточный результат вычисления a. В конце работы алгоритма a примет значение наименьшего неотрицательного вычета $b^n (\mathrm{mod}\, m)$. Пусть $n = n_0 \cdot 2^0 + n_1 \cdot 2^1 + n_2 \cdot 2^2 + \square + n_{k-1} 2^{k-1}$, где n_j, $j = 0, 1, 2, ..., k-1$ – цифры двоичной записи числа n. Каждое n_j равно либо 1, либо 0. Принимаем начальное значение $a = 1$. Первые шаги алгоритма метода повторного возведения в квадрат представлены в табл. 1.6.

Таблица 1.6 – Первые шаги алгоритма

$j = 0$	$b_0 = b (\mathrm{mod}\, m)$	при $n_0 = 1 \Rightarrow a = b_0$
$j = 1$	$b_1 = b^2 (\mathrm{mod}\, m)$	при $n_1 = 1 \Rightarrow a = (a \cdot b_1) \bmod m$
$j = 2$	$b_2 = b_1^2 (\mathrm{mod}\, m)$	при $n_2 = 1 \Rightarrow a = (a \cdot b_2) \bmod m$
$j = 3$	$b_3 = b_2^2 (\mathrm{mod}\, m)$	при $n_3 = 1 \Rightarrow a = (a \cdot b_3) \bmod m$

Алгоритм продолжается для всех $j = 0, 1, 2, ..., k-1$. При $n_j = 0$ достигнутое значение a не меняется. На j-том шаге, получим $b_j = b^{2^j} (\mathrm{mod}\, m)$. Если $n_j = 1$, то есть – когда 2^j входит в двоичное представление числа n, поэтому используем b_j как множитель для вычисления нового значения a и не делаем этого при $n_j = 0$. После выполнения шагов по всем j, получим искомое $a = b^n (\mathrm{mod}\, m)$.

Пример 3. Найти $5^{17} \bmod 7$ методом повторного возведения в квадрат.

Получение бинарного («двоичного») представления числа 17 показано в табл. 1.7. Строки таблицы заполняются справа–налево.

Таблица 1.7 – Построение двоичного представления числа 17

Четное число без остатка					16
Делим на 2	1	2	4	8	17
Остаток (бинарное представление числа)	1	0	0	0	1
Позиция (разряд)	4	3	2	1	0

Проверка: $n = 1 \cdot 2^0 + 1 \cdot 2^4 = 1 + 16 = 17.$

Процедура «повторного возведения в квадрат» для определения $5^{17} \bmod 7$ представлена в табл. 1.8.

Таблица 1.8 – «Повторное возведение в квадрат» для $5^{17} \bmod 7$

j	n_j	b_j	a_j
0	1	$b_0 = 5(\bmod 7) = 5$	$a = b_0 = 5$
1	0	$b_1 = 5^2(\bmod 7) = 25 \bmod 7 =$ $= (3 \cdot 7 + 4) \bmod 7 = 4$	$a = a = 5$
2	0	$b_2 = 4^2(\bmod 7) = 16 \bmod 7 =$ $= (2 \cdot 7 + 2) 16 \bmod 7 = 2$	$a = a = 5$
3	0	$b_3 = 2^2(\bmod 7) = 4 \bmod 7 = 4$	$a = a = 5$
4	1	$b_4 = 4^2(\bmod 7) = 16 \bmod 7 =$ $= (2 \cdot 7 + 2) \bmod 7 = 2$	$a = (a \cdot b_4) \bmod m = (5 \cdot 2) \bmod 7 =$ $= 10 \bmod 7 = (7 + 3) \bmod 7 = 3$

Ответ: $5^{17} \bmod 7 = 3$. Результат совпадает с ответом примера, представленного в задании 1.2.

Примеры индивидуальных заданий для проверки умений по заданию 1.4 приведены в табл. 1.9.

Таблица 1.9 – Примеры задач для проверки умений

№	задание	№	задание	№	задание
1	$3^9(\bmod 5)$	7	$2^{17}(\bmod 5)$	13	$9^3(\bmod 11)$
2	$3^8(\bmod 5)$	8	$2^{16}(\bmod 5)$	14	$9^3(\bmod 11)$
3	$2^{25}(\bmod 7)$	9	$2^8(\bmod 5)$	15	$2^{16}(\bmod 7)$
4	$7^3(\bmod 3)$	10	$3^{17}(\bmod 11)$	16	$7^9(\bmod 3)$
5	$3^8(\bmod 5)$	11	$3^8(\bmod 11)$	17	$7^8(\bmod 5)$
6	$2^{17}(\bmod 7)$	12	$9^8(\bmod 11)$	18	$3^{17}(\bmod 7)$

ПРАКТИЧЕСКОЕ ЗАНЯТИЕ № 2

Операторы шифрования: общие свойства и требования. Совершенная секретность и случайные ключи. Совершенный шифр Вернама. Информационная цена криптозащиты. Избыточность сообщений и её роль в криптоанализе. Оценка расстояния единственности.

Задание 2.1. Провести «вручную» шифрование и дешифровку выражения на естественном языке при помощи методов замены и перестановки:

– метода Цезаря с ключевым словом;

– метода Цезаря с аффинной формулой расчета заменяющего символа;

– метода прямоугольника Плейфера;

– метода двойной перестановки с ключевым словом и цифровым ключом.

Правила применения криптосистем замены и перестановки рассматривается на конкретных примерах при проведении практических занятий, после чего задания выполняются на индивидуальных примерах, самостоятельно сформулированных студентом. Корректность постановки задачи входит в само задание: ключевое слово и шифруемая фраза выбираются студентом самостоятельно и утверждаются преподавателем, проводящим практические занятия.

Задание 2.2. Шифрование при помощи криптосистемы RSA.

Краткое изложение основ и правил применения криптосистемы RSA с привязкой к практическим аспектам рассматриваются при проведении практического занятия на конкретном примере. Алгоритм и расчетные формулы подробно рассматриваются в разделе теоретического обучения (либо на лекционных занятиях, либо при выполнении СРС).

Пример. *Дано*: $p = 35279$, $q = 34361$, $m = "ROSTOV"$.

Этап 1. Генерация ключей

Вычислим n из выражения $n = p \cdot q = 35279 \cdot 34361 = 1212221719$.

Найдем $\varphi(n)$: $\varphi(n) = (p-1) \cdot (q-1) = 1212152080$.

Выберем значение $e = 65537$.

На практике наиболее используемыми вариантами являются значения 17 , 65537 . Каждое из этих чисел содержит в двоичном представлении только две единицы, поэтому алгоритм повторного возведения в квадрат позволяет выполнить шифрование быстро.

Определим значение d :

$$d = e^{-1}\left(\operatorname{mod}\varphi\left(n\right)\right) = 65537^{-1}\left(\operatorname{mod}1212152080\right) = 942299953.$$

Открытый ключ:

Секретный ключ: $(e,n) = \left(65537,1212221719\right)$.

$d = 942299953$.

Этап 2. Преобразование открытого текста в числовой эквивалент

Выполним преобразование открытого текста в числовой эквивалент, поставив в соответствие латинскому алфавиту систему счисления по основанию 26 и выполнив перевод значения открытого текста из данной системы счисления в десятичную систему счисления:

$$M = "ROSTOV" =$$
$$= 17 \cdot 26^5 + 14 \cdot 26^4 + 18 \cdot 26^3 + 19 \cdot 26^2 + 14 \cdot 26^1 + 21 \cdot 26^0 = 208710653.$$

Этап 3. Шифрование

Выполним шифр преобразование:

$$C = M^e\left(\operatorname{mod}n\right) = 208710653^{65537}\left(\operatorname{mod}1212221719\right) = 301941131.$$

Этап 4. Преобразование шифртекста в символьное представление

Преобразуем полученный числовой эквивалент шифртекста

$$C = 301941131 = 25 \cdot 26^5 + 10 \cdot 26^4 + 19 \cdot 26^3 + 4 \cdot 26^2 + 12 \cdot 26^1 + 11 \cdot 26^0 =$$
$$= "ZKTEML".$$

Шифртекст: $C = "ZKTEML"$.

Этап 5. Преобразование символьного представления шифртекста в числовой эквивалент

Выполним преобразование шифртекста в числовую форму:

$$C = "ZKTEML" = 25 \cdot 26^5 + 10 \cdot 26^4 + 19 \cdot 26^3 + 4 \cdot 26^2 + 12 \cdot 26^1 + 11 \cdot 26^0 =$$

$$= 301941131.$$

Этап 6. Дешифрование

Выполним дешифрование:

$$M = C^d \pmod{n} = 301941131^{942299953} \pmod{1212221719} = 208710653.$$

Этап 7. Восстановление символьного представления открытого текста

Преобразуем полученный числовой эквивалент открытого текста

$$M = 208710653 =$$

$$= 17 \cdot 26^5 + 14 \cdot 26^4 + 18 \cdot 26^3 + 19 \cdot 26^2 + 14 \cdot 26^1 + 21 \cdot 26^0 = "ROSTOV".$$

Расшифрованный открытый текст: $M = "ROSTOV"$.

ПРАКТИЧЕСКОЕ ЗАНЯТИЕ №3.

Принципы блочного многораундового шифрования. Схема Фейстеля. Генерирование блочных шифров. Блочный шифр DES, разновидности алгоритма DES. Алгоритм RC6. Особенности отечественного стандарта шифрования ГОСТ 28147-89. Потоковые шифры. Примеры потоковых шифров.

1. Цель работы

Изучить российский стандарт шифрования ГОСТ 28147-89. Получить представление о режимах шифрования, обеспечиваемых стандартом. Определить понятие имитовставки, таблиц подстановки. Проанализировать свойства шифртекстов ГОСТ 28147-89.

2. Порядок работы

Последовательно, в течение отведенного расписанием занятий времени, отработать следующие вопросы:

- изучение теоретического материала;
- формирование файлов ключа и таблиц подстановки;
- зашифрование/расшифрование различных файлов по алгоритмы шифрования ГОСТ 28147-89.
- анализ свойств шифртекстов ГОСТ 28147-89.
- оформить отчет по лабораторной работе и защитить его.

3. Теоретическое введение

Описание стандарта шифрования СССР (позже – Российской Федерации) содержится в документе, озаглавленном «Алгоритм криптографического преобразования ГОСТ 28147-89» [1]. Для дальнейшего детального описания алгоритма и его программной реализации потребуется использовать ряд обозначений, которые вводятся ниже (обозначения приводятся по документам [1] и [2]).

В рассматриваемом описании элементы данных, участвующие в преобразовании (элементы открытого текста или ключа) состоят из нескольких компонент: $X = (X_0, X_1, ..., X_{n-1}) = X_0||X_1||...||X_{n-1}$.

Процедура объединения нескольких элементов данных в один называ-

ется *конкатенацией* данных и обозначается символом ||.

Данные могут интерпретироваться как бинарный вектор длины n:

$X = (x_0, x_1, ..., x_{n-1}) = x_0 + 2^1 \cdot x_1 + ... + 2^{n-1} \cdot x_{n-1}$.

Если над векторами выполняется некоторая логическая операция, то предполагается, что данная операция выполняется над соответствующими битами. $A \bullet B = (a_0 \bullet b_0, a_1 \bullet b_1, ..., a_{n-1} \bullet b_{n-1})$, где $n = |A| = |B|$, т.е. размерность векторов A и B, символом "•" обозначается произвольная логическая операция; в ГОСТе операция «сумма по модулю 2», задаваемая инструкцией XOR процессора Intel или, в другой интерпретации, операция «*исключающего или*».

Далее необходимо указать, что описание алгоритма имеет иерархическую структуру, а именно, описаны три алгоритма нижнего уровня, называемые в тексте [1] *циклами*.

В свою очередь, каждый из базовых циклов представляет собой многократное повторение одной процедуры, называемой далее итерацией.

В ГОСТе ключевая информация состоит из двух структур данных. Помимо собственно *ключа*, являющегося параметром линейного преобразования имеется *таблица замен* – параметр нелинейной компоненты (размешивающего преобразования)

Ключ является массивом из восьми 32-битных векторов: $K = \{K_i\}_{0 \le i \le 7}$. В ГОСТе элементы ключа используются как 32-разрядные целые числа без знака (unsigned long в нотации языка C). Таким образом, размер ключа составляет 32·8=256 бит или 32 байта.

Таблица замен представляется матрицей 8×16, содержащей 4-битовые элементы, которые можно представить в виде чисел от 0 до 15. Строки *таблицы замен* называются *узлами замен*, они должны содержать различные значения, то есть каждый *узел замен* должен содержать 16 различных чисел от 0 до 15 в произвольном порядке, т.е. представляет собой подстановку степени 2^4=16. Таблица замен обозначается символом H: $H = \{H_{i,j}\}_{0 \le j \le 15}^{0 \le i \le 7}, 0 \le H_{i,j} \le 15$. Таким образом, общий объем таблицы замен равен: 8 узлов (подстановок) × 16

элементов/узел × 4 бита/элемент = 512 бит или 64 байта.

Одна итерация криптографического преобразования определяет преобразование 64-битового блока данных (далее рассматриваемого как пара 32-битных векторов). Параметром этого преобразования является 32-битовый блок, в качестве которого используется какой-либо элемент ключа. Схема одной итерации приведена на рисунке 3.1 (все рисунки приведены по [1] и [2]).

Исходные данные для одной итерации:

N – преобразуемый 64-битовый блок данных, в ходе выполнения шага его младшая (N_1) и старшая (N_2) части обрабатываются как отдельные 32-битовые (4-х байтные) целые числа. Таким образом, можно записать $N=(N_1//N_2)$.

$\boldsymbol{X}$ – 32-битовый элемент ключа;

Сложение с ключом. Младшая половина преобразуемого блока складывается по модулю 2^{32} с используемым на шаге элементом ключа, результат передается на следующий шаг;

Поблочная замена. 32-битовое значение, полученное на предыдущем шаге, интерпретируется как массив из восьми 4-битовых блоков (полубайт): $S = (S_0,S_1,S_2,S_3,S_4,S_5,S_6,S_7)$.

Далее значение каждого из восьми блоков заменяется на новое, которое выбирается по таблице согласно определению подстановки: значение блока S_i заменяется на S_i-й по порядку элемент i-го узла замены (т.е. i-й строки таблицы замен). Приводимый ниже рисунок обычно называют криптографической схемой преобразования или кратко, криптосхемой.

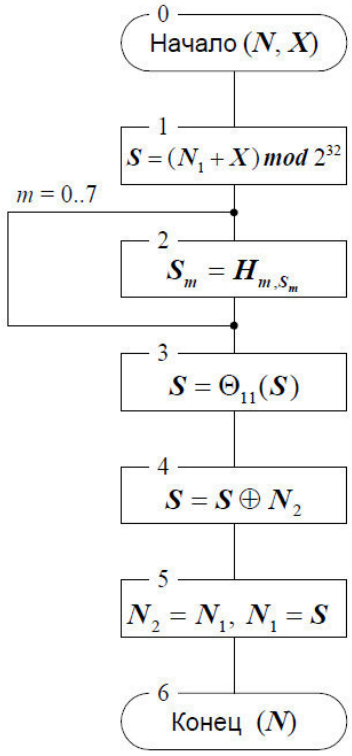


Рисунок 3.1 – Схема одной итерации алгоритма ГОСТ 28147-89.

Циклический сдвиг на 11 бит влево. Результат предыдущего шага сдвигается циклически на 11 бит в сторону старших разрядов и передается на следующий шаг. На схеме алгоритма символом Q_{11} обозначена функция циклического сдвига своего аргумента на 11 бит в сторону старших разрядов.

Побитовое сложение. Значение, полученное на предыдущем шаге, побитно складывается по модулю 2 со старшей половиной преобразуемого блока.

Сдвиг по цепочке. Младшая часть преобразуемого блока сдвигается на место старшей, а на ее место помещается результат выполнения предыдущего шага.

Полученное значение преобразуемого блока возвращается как результат выполнения алгоритма основного шага преобразования.

Здесь необходимо сделать некоторые замечания. Дело в том, что реализация подстановок на полубайтах естественным образом является неоптимальной для существующих процессоров.

Покажем, что существует алгоритм получения «развернутой» подстановки на байтах. Действительно, рассмотрим байт B как конкатенацию полубайтов, $B=b_1||b_2$, b_1 и b_2 принимают значение от 0 до F (в шестнадцатеричной записи). Для реализации узла замены используются две подстановки

$p_1=(p_{10}, p_{11}, ..., p_{1F})$ и $p_2=(p_{20}, p_{21}, ...,p_{2F})$, тогда подстановка

$p_{12}=(p_{10}||p_{20}, p_{10}||p_{21}, ..., p_{10}||p_{2F}, p_{11}||p_{20}, p_{11}||p_{21}, ..., p_{11}||p_{2F}, ..., p_{1F}||p_{20}, ..., p_{1F}||p_{2F})$, реализует отображение целого байта, тождественное независимому применению подстановок p_1 и p_2. Степень подстановки p_{12} равна 256.

Рассмотрим фрагмент кода, который реализует указанное развертыва-

ние.

Пусть

char compuz[8][16];

массив, содержащий 8 узлов замены (подстановки, как было указано, имеют степень 16).

Тогда преобразование пары подстановок (compuz[0] и compuz[1] в развернутый узел fulluz [256] задается операторами

```
for(i=0;i<16;i++)
    for(j=0;j<16;j++) fulluz[i*16+j]=(compuz[0][i]<<4)^compuz[1][j];
```

Далее везде (если не указано особо) будем предполагать, что узлы замены расширены и имеет размер 256 байт (таким образом, совокупность компактных узлов замены по описанному алгоритму преобразуется в 1024 байта расширенных подстановок). Эта операция одновременно является и мерой оптимизации при реализации шифра.

Обратное преобразование развернутого узла замены в исходный можно выполнить следующей парой циклов (предполагаем, что развернутый узел находится в массиве pod):

```
for(i=0;i<16;i++)
    compuz[1][i]= pod[i]&0x0F;
for(i=0;i<16;i++)
    compuz[0][i]=(pod[i*16]&0xF0)>>4;
```

Приведем теперь реализацию одной итерации ГОСТ на языке ассемблера процессора Intel (предполагается, что процессор не менее чем 386). Полагаем, что N_1 и N_2 размещаются в регистрах eax и edx соответственно. Пусть файл uzv.dat содержит развернутые узлы замены (4x256 байт), лежащие последовательно от метки uz0

Включим файл данных, содержащий узел замены.

include uzv.dat

Номер инструкции	Команда	Выполняемые действия
1	2	3
1	gost MACRO	начало макроса
2	mov ecx,eax	сохранение N_1
3	add eax,dword ptr cur_key[I]	сумма по mod 2^{32} с i-m элементом ключа (размером 32 бита)
4	mov bx,offset uz0	подготовка к реализации подстановки на расширенном узле замены
5	xlat	реализация первой подстановки, результат получен в регистре al
6	ror eax,8	перемещение результата вправо на байт, получение в al исходного значения для реализации следующей подстановки
7	inc bh	переход к следующему узлу замены
8	xlat	
9	ror eax,8	
10	inc bh	
11	xlat	
12	ror eax,8	
13	inc bh	
14	xlat	
15	rol eax,3	сдвиг на дополнительные 3 байта (до необходимых 11)
16	xor eax,edx	
17	mov edx,ecx	сдвиг по цепочке
18	ENDM	окончание макроса

Дадим некоторые пояснения к приведенному фрагменту кода. Команда **xlat** процессора Intel представляет собой реализацию подстановки степени 256 (в описании команд процессора она названа «заменой по таблице»), при ее выполнении байт, содержащийся в регистре **al** заменяется на байт, находящийся по адресу **bx+al** *(bx должен содержать адрес подстановки)*. Команда **ror eax,8**, следующая после первых трех команд **xlat** позволяет «выдвигать» в регистр **al** последовательно байты для их преобразования. После четвертой команды **xlat** получаем, что содержимое регистра **eax** уже сдвинуто на байт (8 бит), соответственно, для получения необходимого по описанию сдвига на 11 необходим дополнительный сдвиг на 3 бита. Поскольку развернутые узлы замены расположены в памяти последовательно, то для перехода к следующему узлу необходимо увеличить содержимое регистра bx на 256, что и делается прибавлением единицы к старшему байту регистра bx (команда inc bh).

Приведем реализацию одной итерации на языке С:

```
void elem_gost( unsigned long  *aa, unsigned long  *bb, unsigned long *key)
{
unsigned char r[4];

 *(unsigned long *)r=*aa+*key;
* r   =*(pod+    * r   );
 *(r+1)=*(pod+256+*(r+1));
 *(r+2)=*(pod+512+*(r+2));
 *(r+3)=*(pod+768+*(r+3));
*bb^=(*(unsigned long *)r<<11)|((*(unsigned long *)r>>21)&0x7FF);
}
```

В приводимой процедуре переменная aa соответствует N_1, а bb – N_2, развернутый узел замены помещен в глобальную переменную pod и, как и ранее, представляет собой 4 последовательно расположенные подстановки

степени 256. Key – указатель на массив unsigned long, содержащий ключ криптографического преобразования.

Базовые циклы криптографических преобразований

Базовые циклы алгоритма ГОСТ построены из итераций криптографического преобразования, рассмотренных в предыдущем разделе.

Цикл зашифрования данных использует элементы ключа в следующей последовательности (напомним, что элемент ключа – 4-х байтовое целое число и в ключе таких элементов 8):

K_0,K_1, ..., K_7,K_0,K_1,..., K_7,K_0,K_1,...,K_7,K_7,...,K_1,K_0, или, иначе говоря, 32-битные элементы ключа выбираются 3 раза в прямой и один раз в обратной последовательности (по [2] обозначение 32-З).

Цикл расшифрования данных:

K_0,K_1,...,K_7,K_7,K_6,,...,K_2,K_1,K_0,K_7,K_6,...,K_1,K_0,K_7,K_6,...,K_1,K_0. , т.е. элементы ключа используются один раз в прямой и три раза в обратной последовательности (по [2] обозначение 32-Р).

Здесь уместно заметить, что ГОСТ может использоваться в режиме контроля целостности, или, в приводимой во введении терминологии, в режиме имитозащиты данных или выработки имтовставки.

Цикл выработки имитовставки:

$K_0,K_1,K_2,K_3,K_4,K_5,K_6,K_7,K_0,K_1,K_2,K_3,K_4,K_5,K_6,K_7$.

Цикл расшифрования должен быть обратным циклу зашифрования, то есть последовательное применение этих двух циклов к произвольному блоку данных должно дать в итоге исходный блок.

При реализации криптографических преобразований и особенно при оптимизации тех или иных криптографических процедур возникает задача *проверки соответствия* программной или аппаратной реализации описанию криптографического алгоритма. Как правило, для решения этой задачи используется система тестовых примеров.

Приведем тестовый пример, который позволит проверять правильность реализации ГОСТ.

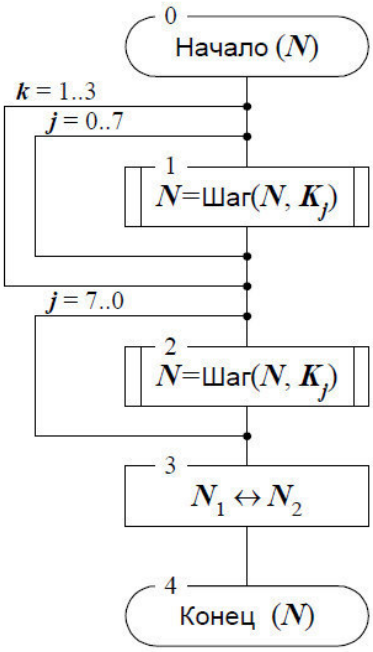


Рисунок 3.2 – Схема цикла зашифрования

Цикл зашифрования на языке Ассемблера (в виде макроса)

```
przam proc near
 REPT 3
 I=0
 REPT 8
 gost
 I=I+4
 ENDM
 ENDM
 I=28
 REPT 8
 gost
 I=I-4
 ENDM
 xchg eax,edx
 ret
przam ENDP
```

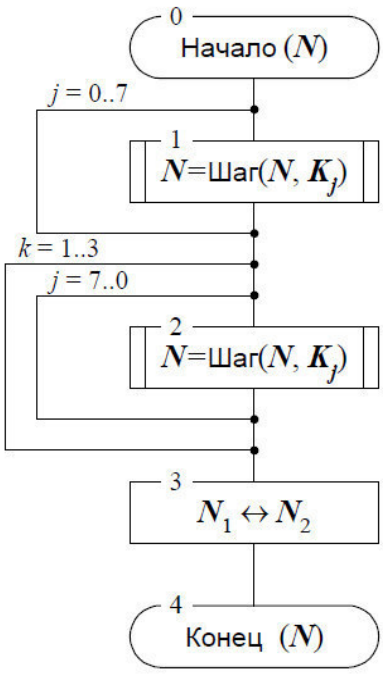


Рисунок 3.3 – Схема цикла расшифрования

Цикл расшифрования

```
przam_1 proc near
 I=0
 REPT 8
 gost
 I=I+4
 ENDM
 REPT 3
 I=28
 REPT 8
 gost
 I=I-4
 ENDM
 ENDM
 xchg eax,edx
 ret
przam_1 ENDP
```

0	0x c	0x 9	0x f	0x e	0x 8	0x 1	0x 3	0x a	0x 2	0x 7	0x 4	0x d	0x 6	0x 0	0x b	0x 5
1	0x 4	0x 2	0x f	0x 5	0x 9	0x 1	0x 0	0x 8	0x e	0x 3	0x b	0x c	0x d	0x 7	0x a	0x 6
2	0x e	0x 9	0x b	0x 2	0x 5	0x f	0x 7	0x 1	0x 0	0x d	0x c	0x 6	0x a	0x 4	0x 3	0x 8
3	0x d	0x 8	0x e	0x c	0x 7	0x 3	0x 9	0x a	0x 1	0x 5	0x 2	0x 4	0x 6	0x f	0x 0	0x b
4	0x 8	0x f	0x 6	0x b	0x 1	0x 9	0x c	0x 5	0x d	0x 3	0x 7	0x a	0x 0	0x e	0x 2	0x 4
5	0x 3	0x e	0x 5	0x 9	0x 6	0x 8	0x 0	0x d	0x a	0x b	0x 7	0x c	0x 2	0x 1	0x f	0x 4
6	0x c	0x 6	0x 5	0x 2	0x b	0x 0	0x 9	0x d	0x 3	0x e	0x 7	0x a	0x f	0x 4	0x 1	0x 8
7	0x 9	0x b	0x c	0x 0	0x 3	0x 6	0x 7	0x 5	0x 4	0x 8	0x e	0x f	0x 1	0x a	0x 2	0x d

то при указании исходного блока в виде двух чисел

0x55555555 0xAAAAAAAA (unsigned long) и ключа

unsigned long Key[8]=

{

0x11111111, 0x22222222, 0x33333333, 0x44444444,

0xFFFFFFFF, 0xFFFFFFFF, 0xFFFFFFFF, 0xFFFFFFFF

};

зашифрованный текст должен получиться таким

0x3113A05E, 0xC4F6F857 (два числа unsigned long)

Цикл зашифрования, реализованный на языке С, будет выглядеть следующим образом (s – указатель на массив из двух блоков открытого текста, предназначенного для зашифрования, k – указатель на массив ключа).

```
void pz(unsigned long *s,unsigned long *k)
{
 void  elem_gost(unsigned long *, unsigned long *,unsigned long *);
 unsigned long  cur;
 elem_gost(s,s+1,k  );
 elem_gost(s+1,s,k+1);
 elem_gost(s,s+1,k+2);
 elem_gost(s+1,s,k+3);
 elem_gost(s,s+1,k+4);
 elem_gost(s+1,s,k+5);
 elem_gost(s,s+1,k+6);
 elem_gost(s+1,s,k+7);
//... Далее указанный блок повторяется еще два раза
//... Затем производится обратная выборка ключа
 elem_gost(s,s+1,k+7);
 elem_gost(s+1,s,k+6);
 elem_gost(s,s+1,k+5);
 elem_gost(s+1,s,k+4);
 elem_gost(s,s+1,k+3);
 elem_gost(s+1,s,k+2);
 elem_gost(s,s+1,k+1);
 elem_gost(s+1,s,k  );
   cur=*s; *s=*(s+1); *(s+1)=cur;
}
```

Цикл выработки имитовставки вдвое короче циклов шифрования, порядок использования ключевых элементов в нем такой же, как и в первых 16 шагах цикла зашифрования.

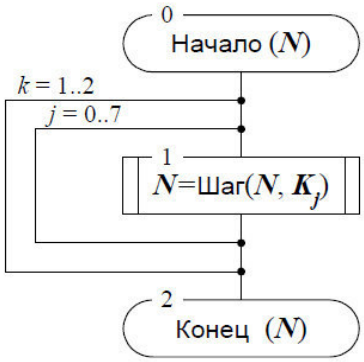


Рисунок 3.4 – Схема цикла выработки имитовставки .

```
imit proc near
 REPT 2
 I=0
 REPT 8
 gost
 I=I+4
 ENDM
 ENDM
ret
imit ENDP
```

Схемы базовых циклов приведены на рисунках 3.2 – 3.4. Каждый из них принимает в качестве аргумента и возвращает в качестве результата 64-битный блок данных, обозначенный на схемах ***N***. Символ Шаг(***N***,***X***) обозначает выполнение основного шага криптопреобразования для блока ***N*** с использованием ключевого элемента ***X***. Между циклами шифрования и вычисления имитовставки есть еще одно отличие – в конце базовых циклов шифрования старшая и младшая часть блока результата меняются местами, это необходимо для обратимости, а для цикла вычисления имитовставки этого не делается.

Основные режимы шифрования

ГОСТ 28147-89 предусматривает три следующих режима шифрования данных:

- простая замена,
- гаммирование,
- гаммирование с обратной связью.

В любом из этих режимов данные обрабатываются блоками по 64 бита, на которые разбивается массив данных, подвергаемый криптографическому преобразованию. Однако в двух режимах гаммирования есть возможность обработки неполного блока данных размером меньше 8 байт, что существенно при шифровании массивов данных с произвольным размером.

Прежде чем перейти к рассмотрению конкретных алгоритмов криптографических преобразований, необходимо пояснить обозначения, используемые на схемах в следующих разделах (напомним, что обозначения приводятся по [2]):

- $T_{\text{о}}, T_{\text{ш}}$ – массивы соответственно открытых и зашифрованных данных;
- $\boldsymbol{T}_i^{\text{о}}, \boldsymbol{T}_i^{\text{ш}}$ – i-тые по порядку 64-битные блоки соответственно открытых и зашифрованных данных: $T_{\text{м}} = (T_1^{\text{"}}, T_2^{\text{"}}, ..., T_n^{\text{"}})$, $T_{\text{щ}} = (T_1^{\text{й}}, T_2^{\text{й}}, ..., T_n^{\text{й}})$, последний блок может быть неполным: $|T_i^{\text{"}}| = |T_i^{\text{щ}}| = 64 \bullet \text{рЦ} 1 \le i < n, 1 \le |T_n^{\text{"}}| = |T_n^{\text{щ}}| \le 64$;
- n – число 64-битных блоков в массиве данных;

$Ц_{\text{Х}}$ – функция преобразования 64-битного блока данных по алгоритму базового цикла «Х»;

Теперь опишем основные режимы шифрования:

Простая замена

Зашифрование в данном режиме заключается в применении цикла зашифрования к блокам открытых данных, расшифрование – цикла расшифрования к блокам зашифрованных данных. Это наиболее простой из режимов, 64-битовые блоки данных обрабатываются в нем независимо друг от друга. Схемы алгоритмов зашифрования и расшифрования в режиме простой замены приведены на рисунках 3.5 и 3.6 соответственно.

Размер массива открытых или зашифрованных данных, подвергаю-

щийся соответственно зашифрованию или расшифрованию, должен быть кратен 64 битам: $|T_о|=|T_ш|=64\cdot n$, после выполнения операции размер полученного массива данных не изменяется.

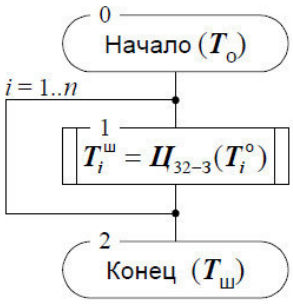


Рисунок 3.5 – Алгоритм зашифрования данных в режиме простой замены

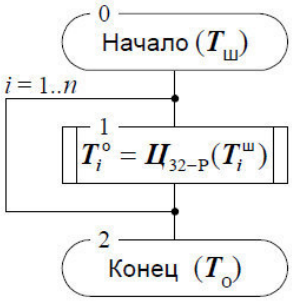


Рисунок 3.6 – Алгоритм расшифрования данных в режиме простой замены.

Гаммирование

Режим гаммирования позволяет получить из блочного шифра поточный, т.е. такой шифр, который позволяет получить последовательность с хорошими статистическим свойствами и использовать ее для наложения на открытый текст с использованием некоторой групповой операции (как правило, коммутативной типа XOR «исключающее или»).

Гаммирование – (в терминологии [1]) это наложение (снятие) на открытые (зашифрованные) данные криптографической гаммы, то есть последова-

тельности элементов данных, вырабатываемых с помощью некоторого криптографического алгоритма, для получения зашифрованных (открытых) данных. Для наложения гаммы при зашифровании и ее снятия при расшифровании в ГОСТе используется операция побитного сложения по модулю 2 (xor).

Теперь перейдем к описанию режима гаммирования. Гамма для этого режима получается следующим образом: с помощью рекуррентного преобразования вырабатываются 64-битные блоки данных, которые далее подвергаются преобразованию по циклу зашифрования в режиме простой замены, в результате получаются блоки гаммы. Благодаря тому, что наложение и снятие гаммы осуществляется при помощи одной и той же операции побитового исключающего или, алгоритмы зашифрования и расшифрования в режиме гаммирования идентичны, их общая схема приведена на рисунке 3.7.

Функция, используемая для выработки гаммы, является рекуррентной функцией: $W_{i+1}=\boldsymbol{f}(W_i)$, где W_i – элементы рекуррентной последовательности, f – функция преобразования. Элемент W_0 является параметром алгоритма для режимов гаммирования, на схемах он обозначен как $\boldsymbol{S}$, и называется в криптографии *синхропосылкой*, а в ГОСТе – *начальным заполнением* одного из регистров. Разработчики ГОСТа решили использовать для инициализации не непосредственно синхропосылку, а результат ее преобразования по циклу зашифрования: $W_0=Ц_{32\text{-}З}(S)$.

Таким образом, последовательность элементов гаммы для использования в режиме гаммирования однозначно определяется ключевыми данными и синхропосылкой. Естественно, для обратимости процедуры шифрования в процессах зашифрования и расшифрования должна использоваться одна и та же синхропосылка. Из требования уникальности (или однократности использования) гаммы (см. также введение, где данный вопрос подробно рассмотрен), невыполнение которого приводит к снижению стойкости шифра, следует, что для шифрования двух различных массивов данных на одном ключе необходимо обеспечить использование различных синхропосылок.

Упомянутая рекуррентная функция имеет следующий вид:

– в 64-битовом блоке старшая и младшая части обрабатываются независимо друг от друга: $\Omega_i = (\Omega_i^0, \Omega_i^1)$, $|\Omega_i^0| = |\Omega_i^1| = 32$, $\Omega_{i+1}^0 = \hat{f}(\Omega_i^0)$, $\Omega_{i+1}^1 = \tilde{f}(\Omega_i^1)$,

– рекуррентные соотношения для старшей и младшей частей следующие:

$\Omega_{i+1}^0 = (\Omega_i^0 + C_1) \bmod 2^{32}$, где $C_1 = 1010101_{16}$;

$\Omega_{i+1}^1 = (\Omega_i^1 + C_2 - 1) \bmod (2^{32} - 1) + 1$, где $C_2 = 1010104_{16}$;

Нижний индекс в записи числа означает его систему счисления.

Второе выражение нуждается в пояснении, так как в тексте ГОСТа [1] приведено другое выражение: $\Omega_{i+1}^1 = (\Omega_i^1 + C_2) \bmod (2^{32} - 1)$, с тем же значением константы $\boldsymbol{C}_2$. Но далее в тексте стандарта дается комментарий, что, оказывается, под операцией взятия остатка по модулю 2^{32}–1 понимается некая специфическая операция. Отличие заключается в том, что согласно ГОСТу $(2^{32}-1) \textit{mod} (2^{32}-1) = (2^{32}-1)$, а не 0. На самом деле, это упрощает реализацию формулы, а математически корректное выражение для нее приведено выше.

Схема алгоритма шифрования в режиме гаммирования приведена на рисунке 3.7, ниже изложены пояснения к схеме.

Определяет исходные данные для основного шага криптопреобразования:

– $T_{о(ш)}$ – массив открытых (зашифрованных) данных произвольного размера, подвергаемый процедуре зашифрования (расшифрования), по ходу процедуры массив подвергается преобразованию порциями по 64 бита;

– S – *синхропосылка*, 64-битный элемент данных, необходимый для инициализации генератора гаммы;

Начальное преобразование синхропосылки, выполняемое для устранения статистических закономерностей используется как начальное состояние рекуррентной функции;

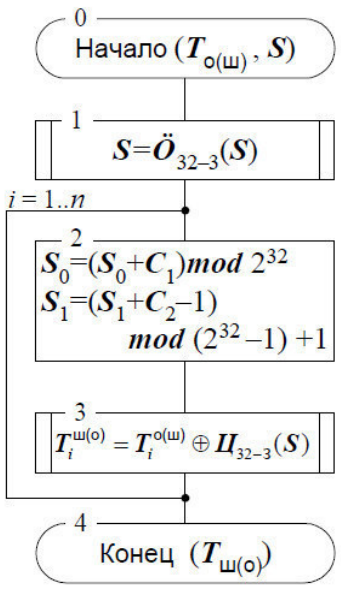


Рисунок 3.7 – Алгоритм зашифрования (расшифрования) данных в режиме гаммирования.

Приведем процедуру реализации шифрования в режиме гаммированиия на языке С (процедура gamm, sp – синхропосылка, din – входной массив данных, dout – выходной массив данных (после наложения гаммы), k – ключ).

```
void gamm(unsigned long *sp,unsigned long *din,unsigned long *dout,int n,unsigned long *k)
{
unsigned long e,d,a[2];
a[0]=*(sp++);
a[1]=*sp;
pz(a,k);
d=a[0];
e=a[1];
while(n-->0)
 {
 a[0]=d+=0x01010101;
a[1]=e=e+0x01010104+((((e>>16)&0xffff)+0x0101)+(((e&0xffff)+0x0104
```

```
)>>16)>>16);
        pz(a,k);
        *(dout++)=a[0]^*(din++);
        *(dout++)=a[1]^*(din++);
        }
       }
```

pz – приводимая выше функция зашифрования в режиме простой замены.

Для проверки соответствия реализации описанному выше алгоритму используйте следующий тестовый пример

```
unsigned long Key01[8]={
0x00000000,0x00000000,0x00000000,0x00000000,
0xFFFFFFFF,0xFFFFFFFF,0xFFFFFFFF,0xFFFFFFFF};
unsigned long
Plain-
text01[4]={0xCCCCCCCC,0x33333333,0x33333333,0xCCCCCCCC};
unsigned long InitVector01[2]={0x11111111,0x22222222};
unsigned long xtext[4];
```

При вызове функции gamm с приведенными параметрами

```
gamm(InitVector01,Plaintext01,xtext,2,Key01);
```

в результате в массиве xtext должно получиться

936a448d 7c85df99

ad763abc 185e2719

Итак, при гаммировании 64-битный элемент, выработанный рекуррентной функцией, подвергается процедуре зашифрования по циклу 32–3, результат используется как элемент гаммы для зашифрования (расшифрования) очередного блока открытых (зашифрованных) данных того же размера.

Результат работы алгоритма – зашифрованный (расшифрованный) массив данных.

Режим гаммирования имеет особенность, которая была отмечена во

введении. В этом режиме биты массива данных шифруются независимо друг от друга. Таким образом, каждый бит шифротекста зависит от соответствующего бита открытого текста и, естественно, порядкового номера бита в массиве: $t_i^{ш} = t_i^{о} \oplus \gamma_i = f(t_i^{о}, i)$. Из этого вытекает, что изменение бита шифротекста на противоположное значение приведет к аналогичному изменению бита открытого текста на противоположный:

$$\bar{t}_i^{ш} = t_i^{ш} \oplus 1 = (t_i^{о} \oplus \gamma_i) \oplus 1 = (t_i^{о} \oplus 1) \oplus \gamma_i = \bar{t}_i^{о} \oplus \gamma_i,$$

где $\bar{t}$ обозначает инвертированное по отношению к t значение бита ($\bar{0} = 1$, $\bar{1} = 0$).

Данное свойство дает злоумышленнику, имеющему доступ к зашифрованному тексту, возможность вносить целенаправленные изменения в соответствующий открытый текст, получаемый после его расшифрования, не обладая при этом секретным ключом. Однако отмеченное свойство режима гаммирования не должно рассматриваться как его недостаток.

Гаммирование с обратной связью

Данный режим похож на режим гаммирования и отличается от него только способом выработки элементов гаммы – очередной элемент гаммы вырабатывается как результат преобразования по циклу 32-З предыдущего блока зашифрованных данных, а для зашифрования первого блока массива данных элемент гаммы вырабатывается как результат преобразования по тому же циклу синхропосылки. Этим достигается так называемое зацепление блоков – каждый блок шифротекста в этом режиме зависит от соответствующего и всех предыдущих блоков открытого текста. Поэтому данный режим иногда называется *гаммированием с зацеплением блоков или гаммированием с зацеплением*. На стойкость шифра зацепление блоков не оказывает существенного влияния.

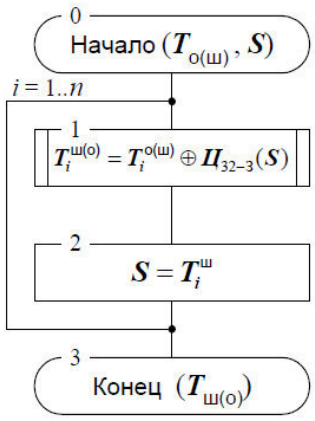


Рисунок 3.8 – Алгоритм зашифрования (расшифрования) данных в режиме гаммирования с обратной связью

Схема алгоритмов за- и расшифрования в режиме гаммирования с обратной связью приведена на рисунке 3.8.

Рассмотрим пример реализации процедуры гаммирования с обратной связью.

// Зашифрование данных в режиме гаммирования с обратной связью

// Возвращаемое значение:

// нет

// Параметры:

// Key - ключ для шифрования

// InData - исходные открытые данные

// Size - длина исходных данных в байтах

// Salt - синхропосылка

// OutData - полученные зашифрованные данные

void GammaEncrypt

Для тестирования данного режима можно использовать следующие данные

static unsigned long Key01[8]=

{

0x00000001,0x00000001,0x00000001,0x00000001,

0xFFFFFFFF,0xFFFFFFFF,0xFFFFFFFF,0xFFFFFFFF};

static unsigned long InitVector01[2]={0xC3A7802A,0x47E3A8FF};

static unsigned long

Plaintext01[4]={0x55555555,0xAAAAAAAA,0x0,0xCCCCCCCC};

GammaEncrypt(Key01,Plaintext01,16,InitVector01,out_d,out_s);

При вызове функции GammaEncrypt в переменной out_d получим значения.

0x807f0572 0x99e6fa4

0xbb42cb56 0x7e336363

Шифрование в режиме гаммирования с обратной связью исключает целенаправленное влияние искажений шифротекста на соответствующий открытый текст. Для сравнения запишем функции расшифрования блока для обоих упомянутых режимов:

$T_i^* = T_i^{щ} \oplus \Gamma_i$, гаммирование;

$T_i^* = T_i^{щ} \oplus ..._{32-}(T_{i-1}^{щ})$, гаммирование с обратной связью;

Если в режиме обычного гаммирования изменения в определенных битах шифротекста влияют только на соответствующие биты открытого текста, то в режиме гаммирования с обратной связью ситуация несколько сложнее. Как видно из соответствующего уравнения, при расшифровании блока данных в режиме гаммирования с обратной связью, блок открытых данных зависит от соответствующего и предыдущего блоков зашифрованных данных. Поэтому, если внести искажения в зашифрованный блок, то после расшифрования искаженными окажутся два блока открытых данных – текущий искаженный и следующий за ним, причем искажения в первом случае будут носить тот же характер, что и в режиме гаммирования, а во втором случае – как в режиме простой замены. Другими словами, в соответствующем блоке открытых данных искаженными окажутся те же самые биты, что и в блоке шифрованных данных, а в следующем блоке открытых данных все биты не-

зависимо друг от друга с вероятностью 1/2 изменят свои значения.

Выработка имитовставки к массиву данных

Рассматривая режимы шифрования, мы отметили влияние искажений в шифртексте на расшифрованный текст. При расшифровании в режиме простой замены соответствующий блок открытых данных оказывается искаженным непредсказуемым образом, а при расшифровании блока в режиме гаммирования изменения предсказуемы. В режиме гаммирования с обратной связью искаженными оказываются два блока, один предсказуемым, а другой непредсказуемым образом. Необходимо учесть, что непредсказуемые изменения в расшифрованном блоке данных могут быть обнаружены только в случае *избыточности* этих данных, причем чем больше степень избыточности, тем вероятнее обнаружение искажения. Большая избыточность имеет место, например, для текстов на естественных и искусственных языках, в этом случае факт искажения обнаруживается с высокой вероятностью. Однако в других случаях, например, при искажении звуковых или графических объектов, мы получим другой образ, искажения в котором могут быть и не обнаружены.

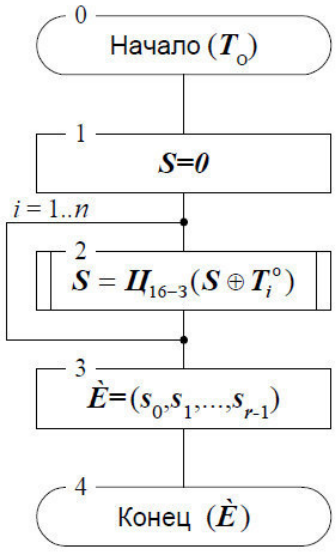


Рисунок 3.9 – Алгоритм выработки имитовставки для массива данных.

Для решения задачи обнаружения искажений в зашифрованном массиве данных с заданной вероятностью в ГОСТе предусмотрен дополнительный режим криптографического преобразования – выработка имитовставки. Имитовставка – это контрольная комбинация, зависящая от открытых данных и секретной ключевой информации. Целью использования имитовставки является обнаружение всех случайных или преднамеренных изменений в массиве информации.

Для потенциального злоумышленника две следующие задачи [2] практически неразрешимы, если он не владеет ключевой информацией:

- вычисление имитовставки для заданного открытого массива информации;
- подбор открытых данных под заданную имитовставку;

Схема алгоритма выработки имитовставки приведена на рисунке 3.9. В качестве имитовставки берется часть блока, полученного на выходе – как правило – 32 младших бита. При выборе размера имитовставки надо принимать во внимание, что вероятность успешного навязывания ложных данных (при неизвестном злоумышленнику ключе) составляет величину порядка 2–L на одну попытку навязывания (L – длина контрольной комбинации в битах). При использовании имитовставки размером 32 бита эта вероятность равна 2–32 (величина порядка 10–9).

4. Практическая часть

Для выполнения практической части работы необходимо сформировать 3 текстовых файла:

а) test1.txt;

б) test2.bmp;

в) test3.exe.

Зашифрованные файлы именовать e*q*_test1.txt, e*q*_test2.bmp, e*q*_test3.exe соответственно. Здесь *q* – номер упражнения.

1. Сформировать файл ключа (двоичный файл длиной 128 бит).
2. С помощью программы gost32.exe зашифровать тестовые файлы на нулевом ключе.
3. С помощью программы gost32.exe зашифровать тестовые файлы на ключе, сформированном на шаге 1.
4. С помощью программы gost32.exe зашифровать тестовые файлы в режиме, обеспечивающем вычисление имитовставки.
5. С помощью программы gost32.exe зашифровать тестовые файлы в режиме гаммирования.
6. С помощью программы gost32.exe зашифровать тестовые файлы в режиме гаммирования с обратной связью.
7. С помощью программы gost32.exe зашифровать тестовые файлы в режиме замены.
8. Определить размер полученных шифртекстов, сравнить его с размером исходных файлов, сделать выводы.
9. Сжать файлы enctest1.txt, enctest2.bmp, enctest3.exe с помощью архиваторов ZIP и RAR. Сравнить размеры полученных файлов с размерами несжатых файлов. Сделать выводы.
10. Оформить отчет по лабораторной работе.

5. Контрольные вопросы

1. Какова мощность ключевого множества ГОСТ 28147-89?
2. Что такое имитовставка? Каково ее назначение? Опишите алгоритм вычисления имитовставки?
3. Перечислите режимы шифрования ГОСТ 28147-89?
4. Каков размер блока в ГОСТ 28147-89?
5. Что такое таблица подстановки? Каково ее назначение? Известны ли Вам аналоги таблиц подстановки?
6. Какой алгоритм, по Вашему мнению, предпочтителен для использования: ГОСТ 28147-89 или AES?

7. Что такое синхропосылка? Каково ее назначение?

6. Рекомендуемая литература

1. Система обработки информации. Защита криптографическая. Алгоритм криптографического преобразования. ГОСТ 28147-89. Госкомстат. М., 1989.

2. Винокуров, А. ГОСТ не прост, а ... очень прост // Монитор. – М. – 1995. – №1.

3. Щербаков, А. К вопросу оптимальной программной реализации криптографических алгоритмов // Безопасность информационных технологий. – М. – 2000. – №2. – с.5 –10.

4. Шнайер, Б. Прикладная криптография. Протоколы, алгоритмы, исходные тексты на языке Си / – М.: Триумф, 2003

ПРАКТИЧЕСКОЕ ЗАНЯТИЕ №4

Основные этапы доступа к ресурсам вычислительной системы; использование простого пароля; использование динамически изменяющегося пароля; взаимная проверка подлинности и другие случаи опознания; способы разграничения доступа к компьютерным ресурсам; разграничение доступа по спискам

Основой любых систем защиты информационных систем (ИС) являются ***идентификация и аутентификация***, так как все механизмы защиты информации рассчитаны на работу с поименованными субъектами и объектами ИС. В качестве субъектов ИС могут выступать как пользователи, так и процессы, а в качестве объектов ИС – информация и другие информационные ресурсы системы.

Присвоение субъектам и объектам личного идентификатора и сравнение его с заданным перечнем называется *идентификацией.* Идентификация обеспечивает выполнение следующих функций:

- установление подлинности и определение полномочий субъекта при его допуске в систему;
- контроль установленных полномочий в процессе сеанса работы;
- регистрация действий пользователя и др.

Аутентификацией (*установлением подлинности)* называется проверка принадлежности субъекту доступа предъявленного им идентификатора и подтверждение его подлинности. Другими словами, аутентификация заключается в проверке: является ли подключающийся субъект тем, за кого он себя выдает.

Общая процедура идентификации и аутентификации пользователя при его доступе в ИС представлена на рисунке 3.1. Если в процессе аутентификации подлинность субъекта установлена, то система защиты информации должна определить его полномочия (совокупность прав). Это необходимо для последующего контроля и разграничения доступа к ресурсам.

Рисунок 3.1 - Классическая процедура идентификации и аутентификации

Способы аутентификации можно разделить на аутентификацию партнеров по общению и аутентификацию источника данных. Аутентификация партнеров по общению используется при установлении (и периодической проверке) соединения во время сеанса. Она служит для предотвращения таких угроз, как маскарад и повтор предыдущего сеанса связи. Аутентификация источника данных – это подтверждение подлинности источника отдельной порции данных.

По направленности аутентификация может быть односторонней (пользователь доказывает свою подлинность системе, например при входе в систему) и двусторонней (взаимной).

Обычно методы аутентификации классифицируют по используемым средствам. В этом случае указанные методы делят на четыре группы:

1. Основанные на знании лицом, имеющим право на доступ к ресурсам системы, некоторой секретной информации – пароля;
2. Основанные на использовании уникального предмета: жетона, электронной карточки и др.;
3. Основанные на измерении биометрических параметров человека – физиологических или поведенческих атрибутах живого организма;
4. Основанные на информации, ассоциированной с пользователем, например, с его координатами.

Рассмотрим эти группы.

Наиболее распространенными простыми и привычными являются методы аутентификации, основанные на *паролях* – секретных идентификаторах субъектов. Здесь при вводе субъектом своего пароля подсистема аутентификации сравнивает его с паролем, хранящимся в базе эталонных данных в зашифрованном виде. В случае совпадения паролей подсистема аутентификации разрешает доступ к ресурсам ИС.

Парольные методы следует классифицировать по степени изменяемости паролей:

- методы, использующие постоянные (многократно используемые) пароли;
- методы, использующие одноразовые (динамично изменяющиеся) пароли.

В большинстве ИС используются многоразовые пароли. В этом случае пароль пользователя не изменяется от сеанса к сеансу в течение установленного администратором системы времени его действительности. Это упрощает процедуры администрирования, но повышает угрозу рассекречивания пароля. Известно множество способов вскрытия пароля: от подсмотра через плечо до перехвата сеанса связи. Вероятность вскрытия злоумышленником пароля повышается, если пароль несет смысловую нагрузку (год рождения, имя девушки), небольшой длины, набран на одном регистре, не имеет ограничений на период существования и т. д. Важно, разрешено ли вводить пароль только в диалоговом режиме или есть возможность обращаться из программы. В последнем случае, возможно запустить программу по подбору паролей – «дробилку».

Более надежный способ – использование одноразовых или динамически меняющихся паролей.

Известны следующие методы парольной защиты, основанные на одноразовых паролях:

- методы модификации схемы простых паролей;
- методы «запрос-ответ»;
- функциональные методы.

В первом случае пользователю выдается список паролей. При аутентификации система запрашивает у пользователя пароль, номер в списке которого определен по случайному закону. Длина и порядковый номер начального символа пароля тоже могут задаваться случайным образом.

При использовании метода «запрос-ответ» система задает пользователю некоторые вопросы общего характера, правильные ответы на которые известны только конкретному пользователю.

Функциональные методы основаны на использовании специальной функции парольного преобразования $f(x)$. Это позволяет обеспечить возможность изменения (по некоторой формуле) паролей пользователя во времени. Указанная функция должна удовлетворять следующим требованиям:

- для заданного пароля x легко вычислить новый пароль $y = f(x)$;

- зная x и y, сложно или невозможно определить функцию $f(x)$.

Наиболее известными примерами функциональных методов являются: метод функционального преобразования и метод «рукопожатия».

Идея метода функционального преобразования состоит в периодическом изменении самой функции $f(x)$. Последнее достигается наличием в функциональном выражении динамически меняющихся параметров, например, функции от некоторой даты и времени. Пользователю сообщается исходный пароль, собственно функция и периодичность смены пароля. Нетрудно видеть, что паролями пользователя на заданных n-периодах времени будут следующие: $x, f(x), f(f(x)), \ldots, f(x)^{n-1}$.

Метод «рукопожатия» состоит в следующем. Функция парольного преобразования известна только пользователю и системе защиты. При входе в ИС подсистема аутентификации генерирует случайную последовательность x, которая передается пользователю. Пользователь вычисляет результат функции $y=f(x)$ и возвращает его в систему. Система сравнивает собственный вычисленный результат с полученным от пользователя. При совпадении указанных результатов подлинность пользователя считается доказанной.

Достоинством метода является то, что передача какой-либо информации, которой может воспользоваться злоумышленник, здесь сведена к минимуму.

В ряде случаев пользователю может оказаться необходимым проверить подлинность другого удаленного пользователя или некоторой ИС, к которой он собирается осуществить доступ. Наиболее подходящим здесь является метод «рукопожатия», так как никто из участников информационного обмена не получит никакой конфиденциальной информации.

Отметим, что методы аутентификации, основанные на одноразовых паролях, также не обеспечивают абсолютной защиты. Например, если злоумышленник имеет возможность подключения к сети и перехватывать передаваемые пакеты, то он может посылать последние как собственные.

В последнее время получили распространение комбинированные методы идентификации, требующие, помимо знания пароля, наличие карточки (token) – специального устройства, подтверждающего подлинность субъекта.

Карточки разделяют на два типа:

- пассивные (карточки с памятью);
- активные (интеллектуальные карточки).

Самыми распространенными являются пассивные карточки с магнитной полосой, которые считываются специальным устройством, имеющим клавиатуру и процессор. При использовании указанной карточки пользователь вводит свой идентификационный номер. В случае его совпадения с электронным вариантом, закодированным в карточке, пользователь получает доступ в систему. Это позволяет достоверно установить лицо, получившее доступ к системе и исключить несанкционированное использование карточки злоумышленником

(например, при ее утере). Такой способ часто называют двухкомпонентной аутентификацией.

Иногда (обычно для физического контроля доступа) карточки применяют сами по себе, без запроса личного идентификационного номера.

К достоинству использования карточек относят то, что обработка аутентификационной информации выполняется устройством чтения, без передачи в память компьютера. Это исключает возможность электронного перехвата по каналам связи.

Недостатки пассивных карточек следующие: они существенно дороже паролей, требуют специальных устройств чтения, их использование подразумевает специальные процедуры безопасного учета и распределения. Их также необходимо оберегать от злоумышленников, и, естественно, не оставлять в устройствах чтения. Известны случаи подделки пассивных карточек.

Интеллектуальные карточки кроме памяти имеют собственный микропроцессор. Это позволяет реализовать различные варианты парольных методов защиты: многоразовые пароли, динамически меняющиеся пароли, обычные запрос-ответные методы. Все карточки обеспечивают двухкомпонентную аутентификацию.

К указанным достоинствам интеллектуальных карточек следует добавить их многофункциональность. Их можно применять не только для целей безопасности, но и, например, для финансовых операций. Сопутствующим недостатком карточек является их высокая стоимость.

Методы аутентификации, основанные на измерении биометрических параметров человека (см. таблицу 3.1), обеспечивают почти 100 % идентификацию, решая проблемы утраты паролей и личных идентификаторов. Однако такие методы нельзя использовать при идентификации процессов или данных (объектов данных), так как они только начинают развиваться (имеются проблемы со стандартизацией и распространением), требуют пока сложного и дорогостоящего оборудования. Это обусловливает их использование пока только на особо важных объектах и системах.

Примерами внедрения указанных методов являются системы идентификации пользователя по рисунку радужной оболочки глаза, отпечаткам ладони, формам ушей, инфракрасной картине капиллярных сосудов, по почерку, по запаху, по тембру голоса и даже по ДНК.

Таблица 3.1 - Примеры методов биометрии

Физиологические методы	**Поведенческие методы**
Снятие отпечатков пальцев Сканирование радужной оболочки глаза Сканирование сетчатки глаза Геометрия кисти руки Распознавание черт лица	Анализ подписи Анализ тембра голоса Анализ клавиатурного почерка

Новым направлением является использование биометрических характеристик в интеллектуальных расчетных карточках, жетонах-пропусках и элементах сотовой связи. Например, при расчете в магазине предъявитель карточки кладет палец на сканер в подтверждение, что карточка действительно его.

Назовем наиболее используемые биометрические атрибуты и соответствующие системы:

- *отпечатки пальцев.* Такие сканеры имеют небольшой размер, универсальны, относительно недороги. Биологическая повторяемость отпечатка пальца составляет 10^{-5} %. В настоящее время пропагандируются правоохранительными органами из-за крупных ассигнований в электронные архивы отпечатков пальцев;

- *геометрия руки.* Соответствующие устройства используются, когда из-за грязи или травм трудно применять сканеры пальцев. Биологическая повторяемость геометрии руки около 2 %;

- *радужная оболочка глаза.* Данные устройства обладают наивысшей точностью. Теоретическая вероятность совпадения двух радужных оболочек составляет 1 из 10^{78};

- *термический образ лица*. Системы позволяют идентифицировать человека на расстоянии до десятков метров. В комбинации с поиском данных по базе данных такие системы используются для опознания авторизованных сотрудников и отсеивания посторонних. Однако при изменении освещенности сканеры лица имеют относительно высокий процент ошибок;

- *голос.* Проверка голоса удобна для использования в телекоммуникационных приложениях. Вероятность ошибки составляет 2 – 5%. Данная технология подходит для верификации по голосу по телефонным каналам связи, она более надежна по сравнению с частотным набором личного номера. Сейчас развиваются направления идентификации личности и его состояния по голосу – возбужден, болен, говорит правду, не в себе и т.д.;

- *ввод с клавиатуры.* Здесь при вводе, например, пароля отслеживаются скорость и интервалы между нажатиями;

- *подпись.* Для контроля рукописной подписи используются дигитайзеры.

Новейшим направлением аутентификации является доказательство подлинности удаленного пользователя по его местонахождению. Данный защитный механизм основан на использовании системы космической навигации, типа GPS (Global Positioning System). Пользователь, имеющий аппаратуру GPS, многократно посылает координаты заданных спутников, находящихся в зоне прямой видимости. Подсистема аутентификации, зная орбиты спутников, может с точностью до метра определить месторасположение пользователя. Высокая надежность аутентификации определяется тем, что орбиты спутников подвержены колебаниям, предсказать которые достаточно трудно. Кроме того, координаты постоянно меняются, что сводит на нет возможность их перехвата. Аппаратура GPS проста и надежна в использовании и сравнительно недорога. Это позволяет ее использовать в случаях, когда авторизованный удаленный пользователь должен находиться в нужном месте.

Суммируя возможности средств аутентификации, ее можно классифицировать по уровню информационной безопасности на три категории:

1. Статическая аутентификация;
2. Устойчивая аутентификация;
3. Постоянная аутентификация.

Первая категория обеспечивает защиту только от несанкционированного доступа (НСД) в системах, где нарушитель не может во время сеанса работы прочитать аутентификационную информацию. Примером средства статической аутентификации являются традиционные постоянные пароли. Их эффективность преимущественно зависит от сложности угадывания паролей и, собственно, от того, насколько хорошо они защищены.

Для компрометации статической аутентификации нарушитель может подсмотреть, подобрать, угадать или перехватить аутентификационные данные и т.д.

Устойчивая аутентификация использует динамические данные аутентификации, меняющиеся с каждым сеансом работы. Реализациями устойчивой аутентификации являются системы, использующие одноразовые пароли и электронные подписи. Усиленная аутентификация обеспечивает защиту от атак, где злоумышленник может перехватить аутентификационную информацию и пытаться использовать ее в следующих сеансах работы. Однако устойчивая аутентификация не обеспечивает защиту от активных атак, в ходе которых маскирующийся злоумышленник может оперативно (в течение сеанса аутентификации) перехватить, модифицировать и вставить информацию в поток передаваемых данных.

Постоянная аутентификация обеспечивает идентификацию каждого блока передаваемых данных, что предохраняет их от несанкционированной модификации или вставки. Примером реализации указанной категории

аутентификации является использование алгоритмов генерации электронных подписей для каждого бита пересылаемой информации.

Парольная система как неотъемлемая составляющая системы защиты информации является частью “переднего края обороны” всей системы безопасности. Поэтому парольная система становится одним из первых объектов атаки при вторжении злоумышленника в защищенную систему.

Для более детального рассмотрения принципов построения парольных систем сформулируем несколько основных определений.

Пароль пользователя - некоторое секретное количество информации, известное только пользователю и парольной системе, которое может быть запомнено пользователем и предъявлено для прохождения процедуры аутентификации. Одноразовый пароль дает возможность пользователю однократно пройти аутентификацию. Многоразовый пароль может быть использован для проверки подлинности повторно.

Учетная запись пользователя-совокупность его идентификатора и его пароля.

База данных пользователей парольной системы содержит учетные записи всех пользователей данной парольной системы.

Под *парольной системой* будем понимать программно-аппаратный комплекс, реализующий системы идентификации и аутентификации пользователей ИС на основе одноразовых или многоразовых паролей. В отдельных случаях парольная система может выполнять ряд дополнительных функций, в частности генерацию и распределение кратковременных (сеансовых) криптографических ключей.

Основными компонентами парольной системы являются:

- интерфейс пользователя;
- интерфейс администратора;
- модуль сопряжения с другими подсистемами безопасности;
- база данных учетных записей.

Некоторые элементы парольной системы (в частности, реализующие интерфейс пользователя) могут быть расположены в местах, открытых для доступа потенциальному злоумышленнику. Поэтому парольная система становится одним из первых объектов атаки при вторжении злоумышленника в защищенную систему. Ниже перечислены типы угроз безопасности парольных систем.

1. Разглашение параметров учетной записи через:

- подбор в интерактивном режиме;
- подсматривание;
- преднамеренную передачу пароля его владельцем другому лицу;
- захват базы данных парольной системы (если пароли не хранятся в базе в открытом виде, для их восстановления может потребоваться подбор или дешифрование);
- перехват переданной по сети информации о пароле;
- хранение пароля в доступном месте;

2. Вмешательство в функционирование компонентов парольной системы через:

- внедрение программных закладок;
- обнаружение и использование ошибок, допущенных на стадии разработки;
- выведение из строя парольной системы.

Некоторые из перечисленных типов угроз связаны с наличием, так называемого, человеческого фактора, проявляющегося в том, что пользователь может:

- выбрать пароль, который легко запомнить и также легко подобрать;
- записать пароль, который сложно запомнить, и положить запись в доступном месте;
- ввести пароль так, что его смогут увидеть посторонние;
- передать пароль другому лицу намеренно или под влиянием заблуждения.

В дополнение к выше сказанному необходимо отметить существование "парадокса человеческого фактора". Заключается он в том, что пользователь нередко стремится выступать скорее противником парольной системы, как, впрочем, и любой системы безопасности, функционирование которой влияет на его рабочие условия, нежели союзником системы защиты, тем самым ослабляя ее. Защита от указанных угроз основывается на ряде перечисленных ниже организационно-технических мер и мероприятий.

Еще одним важным аспектом стойкости парольной системы, является способ хранения паролей в базе данных учетных записей. Возможны следующие варианты хранения паролей:

- в открытом виде;
- в виде свёрток (хеширование);
- зашифрованными на некотором ключе.

Наибольший интерес представляют второй и третий способы, которые имеют ряд особенностей.

Хеширование не обеспечивает защиту от подбора паролей по словарю в случае получения базы данных злоумышленником. При выборе алгоритма хеширования, который будет использован для вычисления сверток паролей, необходимо гарантировать несовпадение значений сверток, полученных на основе различных паролей пользователей. Кроме того, следует предусмотреть механизм, обеспечивающий уникальность сверток в том случае, если два пользователя выбирают одинаковые пароли. Для этого при вычислении каждой свертки обычно используют некоторое количество "случайной" информации, например, выдаваемой генератором псевдослучайных чисел

При шифровании паролей особое значение имеет способ генерации и хранения ключа шифрования базы данных учетных записей. Перечислим некоторые возможные варианты:

- ключ генерируется программно и хранится в системе, обеспечивая возможность ее автоматической перезагрузки;

- ключ генерируется программно и хранится на внешнем носителе, с которого считывается при каждом запуске;

- ключ генерируется на основе выбранного администратором пароля. который вводится в систему при каждом запуске.

Во втором случае необходимо обеспечить невозможность автоматического перезапуска системы, даже если она обнаруживает носитель с ключом. Для этого можно потребовать от администратора подтверждать продолжение процедуры загрузки, например, нажатием клавиши на клавиатуре.

Наиболее безопасное хранение паролей обеспечивается при их хешировании и последующем шифровании полученных сверток, т.е. при комбинации второго и третьего способов.

В большинстве систем пользователи имеют возможность самостоятельно выбирать пароли или получают их от системных администраторов. При этом для уменьшения деструктивного влияния описанного выше человеческого фактора необходимо реализовать ряд требований к выбору и использованию паролей (таблица 3.2).

Таблица 3.2 - Требования к выбору пароля

Требования	**Получаемый эффект**
Установление минимальной длины пароля	Усложняет задачу злоумышленника при попытке подсмотреть пароль или подобрать пароль методом "тотального опробования"
Использование в пароле различных групп символов	Усложняет задачу злоумышленника при попытке подобрать пароль методом "тотального опробования"
Проверка и отбраковка пароля по словарю	Усложняет задачу злоумышленника при попытке подобрать пароль по словарю
Установление максимального срока действия пароля	Усложняет задачу злоумышленника по подбору паролей методом тотального опробования, в том числе без непосредственного обращения к системе защиты (режим off-line)
Установление минимального срока действия пароля	Заменить пароль на старый после его смены по предыдущему требованию
Ведение журнала истории паролей	Обеспечивает дополнительную степень защиты по предыдущему требованию
Применение эвристического алгоритма, бракующего	добрать пароль по словарю или с использованием эвристического алгоритма

пароли на основании данных журнала	
Ограничение числа попыток ввода пароля	Препятствует интерактивному подбору паролей злоумышленником
Поддержка режима принудительной смены пароля пользователя	Обеспечивает эффективность требования, ограничивающего максимальный срок действия пароля
Использование задержки при вводе неправильного пароля	Препятствует интерактивному подбору паролей злоумышленником
Запрет на выбор пароля самим пользователем и автоматическая генерация паролей	Исключает возможность подобрать пароль по словарю. Если алгоритм генерации паролей не известен злоумышленнику, последний может подбирать пароли только методом "тотального опробования"
Принудительная смена пароля при первой регистрации пользователя в системе	Защищает от неправомерных действий системного администратора, имеющего доступ к паролю в момент создания учетной записи

Какой пароль можно однозначно назвать слабым во всех отношениях (за исключением запоминаемости)?

Типичный пример: пароль из небольшого количества (до 5) символов/цифр. По некоторым данным, из 967 паролей одного из взломанных почтовых серверов сети Интернет 335 (почти треть) состояла исключительно из цифр. Количество паролей включающих буквы и цифры оказалось равным 20. Остальные пароли состояли из букв в основном в нижнем регистре за редким исключением (в количестве 2 паролей) включающих спецсимволы (“*”, “_”). Символ “_”, однако, часто встречался в именах пользователей. В 33 случаях имя и пароль пользователя совпадали. Самым популярным оказался пароль 123 (встречался 35 раз, почти каждый 27 пароль). На втором месте пароль qwerty (20 паролей). Как удобно он набирается, не правда ли? Далее следуют: 666 (18 раз), 12 (17 раз), xakep (14 раз) и 1, 11111111, 9128 (по 10 раз). 16 паролей состояли из одного символа/цифры.

Какой же пароль может оказать достойное сопротивление попыткам его подбора? Длинный, состоящий из букв разного регистра, цифр и спецсимволов. При этом он должен быть случайным, т.е. выбор символов осуществляется произвольно (без какой бы то ни было системы) и более нигде не использоваться, при этом единственным местом фиксации пароля должна быть голова единственного человека. Однако необходимо учитывать и вопросы практического использования пароля. Очень длинный пароль сложно запомнить, особенно, если учесть тот факт, что пользователю приходится иметь не один пароль. Осуществить быстрый ввод длинного

пароля также не представляется возможным. Произвольно выбранные символы запомнятся, если их произнесение вслух имеет запоминаемую звуковую форму (благозвучие) или они имеют характерное расположение на клавиатуре, в противном случае без шпаргалки не обойтись.

В случае если пользователю самостоятельно необходимо сформировать пароль, в качестве критериев выбора пароля можно выделить следующие:

- использование букв разных регистров;
- использование цифр и спецсимволов совместно с буквами.

При составлении пароля не рекомендуется использовать:

- свое регистрационное имя в каком бы то ни было виде (как есть, обращенное, заглавными буквами, удвоенное, и т.д);
- свое имя, фамилию или отчество в каком бы то ни было виде;
- имена близких родственников;
- информацию о себе, которую легко можно получить. Она включает номера телефонов, номера лицевых счетов, номер вашего автомобиля, название улицы, на которой вы живете, и т.д.;
- пароль из одних цифр или из одних букв;
- слово, которое можно найти в словарях.

Помочь пользователю составить пароль по определенным критериям могут программы генерации паролей.

Существуют методы количественной оценки стойкости парольных систем (формула Андерсона):

$$4{,}32 \times 10^4 \times (k)\frac{M}{P} \le A^1 \qquad (1.1)$$

где k – количество попыток подбора пароля в минуту;

M – время действия пароля в месяцах;

P – вероятность подбора пароля;

A^1 – мощность пространства (алфавита) паролей (количество символов, которые могут быть использованы при составлении пароля, например, если пароль состоит только из малых английских букв, то A=26, l – длина пароля).

Таким образом, наибольшее влияние на вероятность раскрытия пароля оказывает величина l. Другие составляющие данной формулы чрезвычайно редко оказывают влияние на величину P, превышающее один порядок. Увеличение же длины пароля только на один символ значительно увеличивает требуемое злоумышленнику время для его раскрытия.

Параметры P, V, T и A^1 связаны между собой следующим соотношением:

$$P = \frac{V \times T}{A^1} \qquad (1.2)$$

где P – вероятность подбора пароля в течение его срока действия (подбор осуществляется непрерывно в течение всего срока действия пароля);

V – скорость подбора паролей. Для интерактивного режима определяется как скорость обработки одной попытки регистрации проверяющей стороной. Для режима off-line (на основе свертки пароля) определяется как скорость вычисления значения свертки для одного пробного пароля);

T – срок действия пароля (задает промежуток времени, по истечении которого пароль должен быть сменен);

A^l – мощность пространства паролей (A – мощность алфавита паролей, l – длина пароля).

В случае, когда неизвестна точная длина искомого пароля, максимальное время подбора пароля (T_{max}) будет вычисляться в соответствии со следующей формулой:

$$T_{max} = \frac{\sum_{i=1}^{l} A}{V} \qquad (1.3)$$

Содержание отчета:

- название и цель работы;
- расчет количественной оценки стойкости парольной системы.
- проверка стойкости по времени подбора паролей с помощью специализированных программных средств

Задания

1. Рассчитать оценку стойкости парольной системы защиты для индивидуального варианта. Каждый вариант содержит 2 задания. Первое характерно для ручного подбора паролей, а второе – для подбора с использованием метода перебора, реализуемого с помощью компьютера. Для каждого задания предложить свой, удовлетворяющий требованиям пароль.

Вариант	**V**	**T**
1	15 паролей/мин 1000000 паролей/сек	2 недели 30 дней
2	3 паролей/мин 500000 паролей/сек	10 дней 1 год
3	10 паролей/мин 10000000 паролей/сек	5 дней 40 дней

4	11 паролей/мин 50000 паролей/сек	6 дней 90 дней
5	100 паролей/день 10000 паролей/сек	12 дней 10 дней
6	10 паролей/день 5000000 паролей/сек	1 месяц 150 дней
7	20 паролей/мин 20000 паролей/сек	3 недели 10 дней
8	15 паролей/мин 1000 паролей/сек	20 дней 3 дня
9	3 паролей/мин 10000000 паролей/сек	15 дней 120 дней
10	10 паролей/мин 100000000 паролей/сек	1 неделя 1 год
11	11 паролей/мин 300000 паролей/сек	2 недели 100 дней
12	100 паролей/день 70000 паролей/сек	10 дней 20 дней
13	10 паролей/день 30000000 паролей/сек	5 дней 200 дней
14	20 паролей/мин 80000 паролей/сек	6 дней 60 дней
15	15 паролей/мин 500000000 паролей/сек	12 дней 5 лет
16	3 паролей/мин 10000000 паролей/сек	1 месяц 10 лет
17	10 паролей/мин 50000000 паролей/сек	3 недели 100 лет
18	11 паролей/мин 1000000 паролей/сек	20 дней
19	100 паролей/день 20000 паролей/сек	15 дней 15 дней
20	10 паролей/день 20000000 паролей/сек	1 неделя 365 дней

Занесите проведенные расчеты в отчет по лабораторной работе и **сделайте соответствующие выводы**.

2. В настоящее время существует целый класс программ, позволяющих проверить стойкость пароля любой сложности. Очень удобно пользоваться онлайн программами, которые не требуют загрузки и установки. Для исследования используем программы, расположенные по следующим адресам:

https://password.kaspersky.com/ru/

https://2ip.online/passcheck/

В качестве индивидуального задания предлагается самостоятельно создать несколько категорий паролей:

- пароль, состоящий только из цифр;
- пароль, состоящий только из цифр и букв;
- пароль, состоящий из цифр и букв с различным регистром;
- пароль состоящий из цифр, букв с различным регистром и специальных символов.

По каждой категории паролей предложить 4 разновидности:

состоящий из 5-ти знаков;

состоящий из 7-ми знаков;

состоящий из 9-ти знаков;

состоящий из 11-ти знаков;

С помощью перечисленных выше программных средств провести исследование стойкости паролей для каждой категории и после исследования паролей в каждой из онлайн программ занести полученные данные в таблицу. Пример такой таблицы для программы по адресу https://password.kaspersky.com/ru/ представлен в таблице 3.

Таблица 3 – Исследование стойкости паролей с использованием программы по адресу https://password.kaspersky.com/ru/

Длина пароля	Время подбора пароля на компьютере			
	5 знаков	7 знаков	9 знаков	11 знаков
пароль, состоящий только из цифр				
пароль, состоящий только из цифр и букв				
пароль, состоящий из цифр и букв с различным регистром				
пароль состоящий из цифр, букв с различным регистром и специальных символов				

Такую же таблицу

3. Проведите в интернете поиск аналогичной онлайн программы и проведите 3-е исследование для усреднения полученных результатов. **Сделайте соответствующие выводы**.

Контрольные вопросы

1. Что такое идентификация?
2. Что понимается под аутентификацией?
3. Чем определяется стойкость подсистемы идентификации и аутентификации?
4. Перечислите минимальные требования к выбору пароля.
5. Почему парольная защита с динамически изменяющимся паролем выше, чем с долговременным паролем?
6. Аспекты классификации методов идентификации и аутентификации?
7. Как определить вероятность подбора пароля злоумышленником в течение срока его действия?
8. Выбором, каких параметров можно повлиять на уменьшение вероятности подбора пароля злоумышленником при заданной скорости подбора пароля злоумышленником и заданном сроке действия пароля?

ПРАКТИЧЕСКОЕ ЗАНЯТИЕ №5.

Базовые этапы построения системы комплексной защиты вычислительных систем; анализ моделей нарушителя; угрозы информационно-программному обеспечению вычислительных систем и их классификация

Информационная безопасность в широком смысле – это совокупность средств защиты информации от случайного или преднамеренного воздействия. Независимо от того, что лежит в основе воздействия: естественные факторы или причины искусственного характера – владелец информации несет убытки.

Принципы информационной безопасности

- **Целостность** информационных данных означает способность информации сохранять изначальный вид и структуру как в процессе хранения, как и после неоднократной передачи. Вносить изменения, удалять или дополнять информацию вправе только владелец или пользователь с легальным доступом к данным.
- **Конфиденциальность** – характеристика, которая указывает на необходимость ограничить доступа к информационным ресурсам для определенного круга лиц. В процессе действий и операций информация становится доступной только пользователям, который включены в информационные системы и успешно прошли идентификацию.
- **Доступность** информационных ресурсов означает, что информация, которая находится в свободном доступе, должна предоставляться полноправным пользователям ресурсов своевременно и беспрепятственно.
- **Достоверность** указывает на принадлежность информации доверенному лицу или владельцу, который одновременно выступает в роли источника информации.

Обеспечение и поддержка информационной безопасности включают комплекс разноплановых мер, которые предотвращают, отслеживают и устраняют несанкционированный доступ третьих лиц. Меры ИБ направлены также на защиту от повреждений, искажений, блокировки или копирования информации. Принципиально, чтобы все задачи решались одновременно, только тогда обеспечивается полноценная, надежная защита.

Угрозой информации называют потенциально возможное влияние или воздействие на автоматизированную систему с последующим нанесением убытка чьим-то потребностям.

На сегодня существует более 100 позиций и разновидностей угроз информационной системе. Важно проанализировать все риски с помощью разных методик диагностики. На основе проанализированных показателей с их детализацией можно грамотно выстроить систему защиты от угроз в информационном пространстве.

Классификация уязвимостей систем безопасности

Угрозы информационной безопасности проявляются не самостоятельно, а через возможное взаимодействие с наиболее слабыми звеньями системы защиты, то есть через факторы уязвимости. Угроза приводит к нарушению деятельности систем на конкретном объекте-носителе.

Основные уязвимости возникают по причине действия следующих факторов:

- несовершенство программного обеспечения, аппаратной платформы;
- разные характеристики строения автоматизированных систем в информационном потоке;
- часть процессов функционирования систем является неполноценной;
- неточность протоколов обмена информацией и интерфейса;

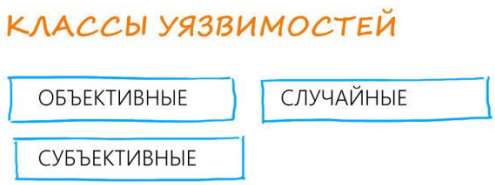
- сложные условия эксплуатации и расположения информации.

Чаще всего источники угрозы запускаются с целью получения незаконной выгоды вследствие нанесения ущерба информации. Но возможно и случайное действие угроз из-за недостаточной степени защиты и массового действия угрожающего фактора.

Существует разделение уязвимостей по классам, они могут быть:

- объективными;
- случайными;
- субъективными.

Если устранить или как минимум ослабить влияние уязвимостей, можно избежать полноценной угрозы, направленной на систему хранения информации.



Объективные уязвимости

Этот вид напрямую зависит от технического построения оборудования на объекте, требующем защиты, и его характеристик. Полноценное избавление от этих факторов невозможно, но их частичное устранение достигается с помощью инженерно-технических приемов, следующими способами:

1. Связанные с техническими средствами излучения:

- электромагнитные методики (побочные варианты излучения и сигналов от кабельных линий, элементов техсредств);
- звуковые варианты (акустические или с добавлением вибросигналов);
- электрические (проскальзывание сигналов в цепочки электрической сети, по наводкам на линии и проводники, по неравномерному распределению тока).

2. Активизируемые:

- вредоносные ПО, нелегальные программы, технологические выходы из программ, что объединяется термином «программные закладки»;
- закладки аппаратуры – факторы, которые внедряются напрямую в телефонные линии, в электрические сети или просто в помещения.

3. Те, что создаются особенностями объекта, находящегося под защитой:

- расположение объекта (видимость и отсутствие контролируемой зоны вокруг объекта информации, наличие вибро- или звукоотражающих элементов вокруг объекта, наличие удаленных элементов объекта);
- организация каналов обмена информацией (применение радиоканалов, аренда частот или использование всеобщих сетей).

4. Те, что зависят от особенностей элементов-носителей:

- детали, обладающие электроакустическими модификациями (трансформаторы, телефонные устройства, микрофоны и громкоговорители, катушки индуктивности);
- вещи, подпадающие под влияние электромагнитного поля (носители, микросхемы и другие элементы).

Случайные уязвимости

Эти факторы зависят от непредвиденных обстоятельств и особенностей окружения информационной среды. Их практически невозможно предугадать в информационном пространстве, но важно быть готовым к их быстрому устранению. Устранить такие неполадки

можно с помощью проведения инженерно-технического разбирательства и ответного удара, нанесенного угрозе информационной безопасности:

1. Сбои и отказы работы систем:

- вследствие неисправности технических средств на разных уровнях обработки и хранения информации (в том числе и тех, что отвечают за работоспособность системы и за контроль доступа к ней);
- неисправности и устаревания отдельных элементов (размагничивание носителей данных, таких как дискеты, кабели, соединительные линии и микросхемы);
- сбои разного программного обеспечения, которое поддерживает все звенья в цепи хранения и обработки информации (антивирусы, прикладные и сервисные программы);
- перебои в работе вспомогательного оборудования информационных систем (неполадки на уровне электропередачи).

2. Ослабляющие информационную безопасность факторы:

- повреждение коммуникаций вроде водоснабжения или электроснабжения, а также вентиляции, канализации;
- неисправности в работе ограждающих устройств (заборы, перекрытия в здании, корпуса оборудования, где хранится информация).

Субъективные уязвимости

Этот подвид в большинстве случаев представляет собой результат неправильных действий сотрудников на уровне разработки систем хранения и защиты информации. Поэтому устранение таких факторов возможно при помощи методик с использованием аппаратуры и ПО:

1. Неточности и грубые ошибки, нарушающие информационную безопасность:

- на этапе загрузки готового программного обеспечения или предварительной разработки алгоритмов, а также в момент его использования (возможно во время ежедневной эксплуатации, во время ввода данных);
- на этапе управления программами и информационными системами (сложности в процессе обучения работе с системой, настройки сервисов в индивидуальном порядке, во время манипуляций с потоками информации);
- во время пользования технической аппаратурой (на этапе включения или выключения, эксплуатации устройств для передачи или получения информации).

2. Нарушения работы систем в информационном пространстве:

- режима защиты личных данных (проблему создают уволенные работники или действующие сотрудники в нерабочее время, они получают несанкционированный доступ к системе);
- режима сохранности и защищенности (во время получения доступа на объект или к техническим устройствам);
- во время работы с техустройствами (возможны нарушения в энергосбережении или обеспечении техники);
- во время работы с данными (преобразование информации, ее сохранение, поиск и уничтожение данных, устранение брака и неточностей).

Ранжирование уязвимостей

Каждая уязвимость должна быть учтена и оценена специалистами. Поэтому важно определить критерии оценки опасности возникновения угрозы и вероятности поломки или обхода защиты информации. Показатели подсчитываются с помощью применения ранжирования. Среди всех критериев выделяют три основных:

- **Доступность** – это критерий, который учитывает, насколько удобно источнику угроз использовать определенный вид уязвимости, чтобы нарушить информационную

безопасность. В показатель входят технические данные носителя информации (вроде габаритов аппаратуры, ее сложности и стоимости, а также возможности использования для взлома информационных систем неспециализированных систем и устройств).

- **Фатальность** – характеристика, которая оценивает глубину влияния уязвимости на возможности программистов справиться с последствиями созданной угрозы для информационных систем. Если оценивать только объективные уязвимости, то определяется их информативность – способность передать в другое место полезный сигнал с конфиденциальными данными без его деформации.
- **Количество** – характеристика подсчета деталей системы хранения и реализации информации, которым присущ любой вид уязвимости в системе.

Каждый показатель можно рассчитать как среднее арифметическое коэффициентов отдельных уязвимостей. Для оценки степени опасности используется формула. Максимальная оценка совокупности уязвимостей – 125, это число и находится в знаменателе. А в числителе фигурирует произведение из КД, КФ и КК.

РАСЧЕТ СТЕПЕНИ ОПАСНОСТИ

$$K(O) = (КД * КФ * КК) / 125$$

Чтобы узнать информацию о степени защиты системы точно, нужно привлечь к работе аналитический отдел с экспертами. Они произведут оценку всех уязвимостей и составят информационную карту по пятибалльной системе. Единица соответствует минимальной возможности влияния на защиту информации и ее обход, а пятерка отвечает максимальному уровню влияния и, соответственно, опасности. Результаты всех анализов сводятся в одну таблицу, степень влияния разбивается по классам для удобства подсчета коэффициента уязвимости системы.

Какие источники угрожают информационной безопасности?

Если описывать классификацию угроз, которые обходят защиту информационной безопасности, то можно выделить несколько классов. Понятие классов обязательно, ведь оно упрощает и систематизирует все факторы без исключения. В основу входят такие параметры, как:

1. Ранг преднамеренности совершения вмешательства в информационную систему защиты:

- угроза, которую вызывает небрежность персонала в информационном измерении;
- угроза, инициатором которой являются мошенники, и делают они это с целью личной выгоды.

2. Характеристики появления:

- угроза информационной безопасности, которая провоцируется руками человека и является искусственной;
- природные угрожающие факторы, неподконтрольные информационным системам защиты и вызывающиеся стихийными бедствиями.

3. Классификация непосредственной причины угрозы. Виновником может быть:

- человек, который разглашает конфиденциальную информацию, орудуя с помощью подкупа сотрудников компании;
- природный фактор, приходящий в виде катастрофы или локального бедствия;
- программное обеспечение с применением специализированных аппаратов или внедрение вредоносного кода в техсредства, что нарушает функционирование системы;

- случайное удаление данных, санкционированные программно-аппаратные фонды, отказ в работе операционной системы.

4. Степень активности действия угроз на информационные ресурсы:

- в момент обрабатывания данных в информационном пространстве (действие рассылок от вирусных утилит);
- в момент получения новой информации;
- независимо от активности работы системы хранения информации (в случае вскрытия шифров или криптозащиты информационных данных).

Существует еще одна классификация источников угроз информационной безопасности. Она основана на других параметрах и также учитывается во время анализа неисправности системы или ее взлома. Во внимание берется несколько показателей.

Классификация угроз

Состояние источника угрозы	• в самой системе, что приводит к ошибкам в работе и сбоям при реализации ресурсов АС; • в пределах видимости АС, например, применение подслушивающей аппаратуры, похищение информации в распечатанном виде или кража записей с носителей данных; • мошенничество вне зоны действия АС. Случаи, когда информация захватывается во время прохождения по путям связи, побочный захват с акустических или электромагнитных излучений устройств.
Степень влияния	• активная угроза безопасности, которая вносит коррективы в структуру системы и ее сущность, например, использование вредоносных вирусов или троянов; • пассивная угроза – та разновидность, которая просто ворует информацию способом копирования, иногда скрытая. Она не вносит своих изменений в информационную систему.
Возможность доступа сотрудников к системе программ или ресурсов	• вредоносное влияние, то есть угроза информационным данным может реализоваться на шаге доступа к системе (несанкционированного); • вред наносится после согласия доступа к ресурсам системы.
Способ доступа к основным ресурсам системы	• применение нестандартного канала пути к ресурсам, что включает в себя несанкционированное использование возможностей операционной системы; • использование стандартного канала для открытия доступа к ресурсам, например, незаконное получение паролей и других параметров с дальнейшей маскировкой под зарегистрированного в системе пользователя.
Размещение информации в системе	• вид угроз доступа к информации, которая располагается на внешних устройствах памяти, вроде несанкционированного копирования информации с жесткого диска; • получение доступа к информации, которая показывается терминалу, например, запись с видеокамер терминалов; • незаконное проникание в каналы связи и подсоединение к ним с целью получения конфиденциальной информации или для подмены реально существующих фактов под видом зарегистрированного сотрудника. Возможно распространение дезинформации; • проход к системной области со стороны прикладных программ и считывание всей информации.

Не стоит забывать о таких угрозах, как случайные и преднамеренные. Исследования доказали, что в системах данные регулярно подвергаются разным реакциям на всех стадиях цикла обработки и хранения информации, а также во время функционирования системы.

В качестве источника случайных реакций выступают такие факторы, как:

- сбои в работе аппаратуры;
- периодические шумы и фоны в каналах связи из-за воздействия внешних факторов (учитывается пропускная способность канала, полоса пропуска);
- неточности в программном обеспечении;
- ошибки в работе сотрудников или других служащих в системе;
- специфика функционирования среды Ethernet;
- форс-мажоры во время стихийных бедствий или частых отключений электропитания.

ПРАКТИЧЕСКОЕ ЗАНЯТИЕ №6.

Функции хеширования: ГОСТ Р34.11-94, MD5, SHA

1. ГОСТ Р 34.11-94.

Изучение криптографических хэш-функций начнем с рассмотрения российского стандарта ГОСТ Р 34.11-94, который описан в нормативном документе «ГОСТ Р 34.11-94. Информационная технология. Криптографическая защита информации. Функция хэширования». Данный стандарт был принят и введен в действие 23 мая 1994 года и действует по настоящий день.

ГОСТ Р 34.11-94 построен по первой из двух рассмотренных схем и использует для преобразования входных данных отечественный стандарт шифрования ГОСТ 28147-89. При этом длина выходного хэш-значения и длина блоков, на которые разбивается хэшируемое сообщение, составляют 256 бит.

Для определения алгоритма хэширования ГОСТ Р 34.11-94 необходимо описать алгоритм вычисления шаговой функции хэширования и итеративную процедуру вычисления значения хэш-функции.

Алгоритм вычисления шаговой функции хэширования, которую обозначим $\kappa(X_1, X_2)$, принимает на вход две 256-битовых строки и возвращает на выходе одну 256-битовую строку.

Он состоит из следующих этапов:

1. Генерация ключей.

Из 256-битовых строк X_1 и X_2 с использованием определенных в стандарте констант генерируются 4 256-битовых ключа.

2. Шифрующее преобразование.

Строка X_1 разбивается на 4 64-битовых подблока, каждый из которых зашифровывается по алгоритму ГОСТ 28147-89 на одном из сгенерированных на предыдущем шаге ключей.

3. Перемешивающее преобразование результата зашифрования.

На данном этапе осуществляется перемешивание полученной 256-битовой строки с применением регистра сдвига.

Теперь введем некоторые обозначения.

M – часть исходной последовательности M, еще не прошедшая обработку.

H – текущее значение хэш-функции.

$\sum$ – текущее значение контрольной суммы.

L – текущее значение длины уже обработанной части исходной последовательности.

0^n – битовая строка длины n, состоящая из нулей.

[+] – операция сложения по модулю 2^{256}.

Алгоритм хэширования ГОСТ Р 34.11-94 состоит из следующих этапов:

1. Присвоить начальные значения

$M \leftarrow M$,

$H \leftarrow H$,

$\Sigma \rightarrow 0^{256}$,

$L \leftarrow 0^{256}$.

2. Проверить условие $|M| > 256$ (необработанная часть входной последовательности содержит более одного блока).

При положительном исходе перейти к этапу 3 для обработки очередного блока входной последовательности.

При отрицательном исходе выполнить вычисления для получения хэш-значения.

$L \leftarrow (L + |M|) \bmod 2^{256}$,

$M' \leftarrow 0^{256-|M|} \| M$,

$\Sigma \leftarrow \Sigma[+]M'$,

$H \leftarrow \kappa(M', H)$,

$H \leftarrow \kappa(L, H)$,

$H \leftarrow \kappa(\Sigma, H)$.

Конец работы алгоритма.

3. Вычислить 256-битовую подстроку M_S строки M $(M = M_P \| M_S)$.

$H \leftarrow \kappa(M_S, H)$,

$L \leftarrow (L + 256) \bmod 2^{256}$,

$\Sigma \leftarrow \Sigma[+]M_S$,

$M = M_P$.

Перейти к этапу 2.

Значение H, полученное на 2-м этапе содержит значение хэш-функции ГОСТ Р 34.11-94.

2. MD5

В отличие от единственного отечественного стандарта хэширования ГОСТ Р 34.11-94 за рубежом и, прежде всего, в США, широкое применение находят несколько хэш-функций. Наиболее популярными являются MD5, SHA-1, а также SHA-2.

MD5 является представителем целого семейства хэш-функций, разработанных Роном Райвестом. Первой их них была функция хэширования MD4 (Message Digest 4), разработанная и опубликованная в 1990 году, улучшенной версией которой является MD5. Кроме того, данное семейство включает в себя созданный в 1992 году алгоритм хэширования MD2 и недавнюю разработку – алгоритм MD6, созданный, в свою очередь, в 2008 году.

SHA-1 (Secure Hash Algorithm 1) – это алгоритм хэширования, созданный Агентством национальной безопасности США совместно с Национальным институтом стандартов и технологий, и опубликованный в 1995 году в качестве стандарта. SHA-2 – вторая версия «безопасного алгоритма хэширования», опубликованная в 2002 году. Данные функции хэширования построены на основе тех же принципов, что и MD5, поэтому их также можно рассматривать в качестве представителей этого семейства.

Рассмотрим хэш-функцию MD5, которая достаточно широко используется на сегодняшний день (например, для безопасного хранения паролей в Windows и Linux).

MD5 принимает на входе битовую последовательность произвольной длины, обрабатывает ее блоками по 512 бит и возвращает на выходе 128-битовое хэш-значение.

Алгоритм преобразований выглядит следующим образом.

1. Добавление битов заполнения.

Исходная последовательность дополняется некоторым количеством битов, чтобы ее битовая длина была сравнима с 448 по модулю 512. Для этого в конец последовательности дописывается единичный бит, а затем нужное количество нулевых битов.

Добавление битов происходит, даже если длина сообщения уже сравнима с 448 по модулю 512.

2. Добавление длины последовательности.

Длина исходной последовательности представляется 64-битовым значением и дописывается в конец последовательности, полученной на первом шаге. Если длина последовательности превосходит $2^{64} - 1$, то учитываются только младшие 64 бита.

Теперь последовательность содержит целое число 512-битовых блоков.

3. Инициализация буфера.

Для хранения результатов промежуточных вычислений используются 4 32-битовых слова A, B, C, D, в которые в начале записываются следующие значения:

A = 0x01234567,

B = 0x89abcdef,

C = 0xfedcba98,

D = 0x76543210.

Начальное состояние (A, B, C, D) называется инициализирующим вектором.

Шаговая функция хэширования MD5 представлена на рисунке 5.

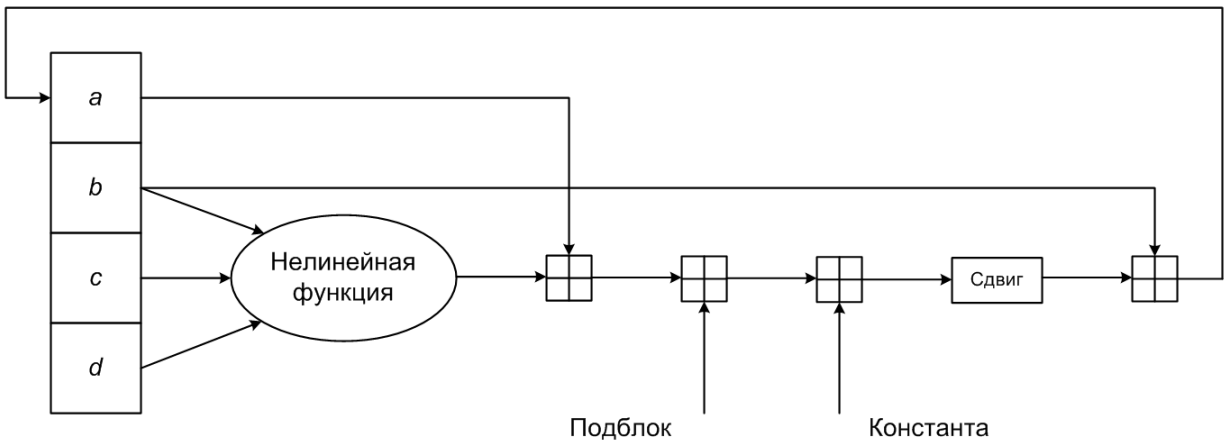


Рисунок 5 – Шаговая функция хэширования MD5

Каждый 512-битовый блок разбивается на 16 32-битовых подблоков и обрабатывается в 4 этапа. На каждом этапе три из четырех переменных *a*, *b*, *c*, *d* преобразуются посредством одной из заданных нелинейных функций, далее следует ряд операций, одна из которых заключается в сложении текущего результата преобразований с очередным подблоком.

Таким образом, шаговая функция хэширования повторяется 64 для каждого блока входной последовательности данных.

5. Асимметричные криптосистемы

Криптосистемы с открытым ключом, также называемые асимметричными криптосистемами, используют два разных ключа: открытый ключ зашифрования и закрытый ключ расшифрования. Открытый ключ в общем случае доступен всем желающим, а закрытый ключ известен только законному владельцу. Оба ключа связаны между собой некоторой зависимостью. При этом данная зависимость такова, что, зная один ключ, вычислить другой практически невозможно.

Криптографические алгоритмы с открытым ключом используются для шифрования данных, для решения проблемы распределения секретных ключей, для выработки цифровой подписи.

С проблемой распределения ключей шифрования сталкиваются все симметричные криптосистемы. Стороны, участвующие в защищенном информационном обмене, должны каким-то образом предварительно получить общий секретный ключ, который нельзя передавать в открытом виде. Кроме того, необходимо обеспечить эффективное управление сеансовыми ключами, безопасное хранение долговременных ключей и т. д.

Все перечисленные проблемы настолько важны, что в современной криптографии управление ключами выделяют в отдельный раздел.

Существует несколько решений задачи предварительного распределения секретных ключей.

1) Физическое распределение. Ключи рассылаются с помощью курьера с той или иной степенью охраны.

2) Распределение с помощью протоколов с секретными ключами. Между пользователями и неким доверенным центром распределены долговременные ключи. Генерирование сеансовых ключей и обмен ими между пользователями происходит под управлением доверенного центра. Однако, долговременные ключи могут быть переданы только физическим путем.

3) Распределение с помощью протоколов с открытым ключом. Это наиболее важное приложение криптографии с открытым ключом.

Базовым понятием в криптографии с открытым ключом является понятие однонаправленной или односторонней функции. Значение такой функции для заданного аргумента достаточно легко вычислить, но практически невозможно вычислить значение аргумента для заданного значения функции. Непосредственно для шифрования однонаправленные функции не используются. Для шифрования применяются однонаправленные функции с лазейкой (ловушкой). Для такой функции вычислить обратную функцию достаточно просто, если известна некоторая секретная информация, и практически невозможно, если эта информация неизвестна.

Основными задачами, приводящими к однонаправленным функциям, являются задачи разложения на множители и дискретного логарифмирования в конечном поле.

В асимметричных криптосистемах, построенных на однонаправленных функциях с лазейкой, открытый ключ определяет конкретную реализацию функции, а закрытый ключ дает информацию о лазейке.

1. Протокол Диффи-Хеллмана

Концепция криптографии с открытым ключом была разработана Диффи и Хеллманом и представлена ими в статье «Новые направления в криптографии», опубликованной в 1976 году. Эта статья содержала саму постановку задачи, но ее решение

было только частичным. Диффи и Хеллман не смогли разработать асимметричную криптосистему, но смогли решить проблему распределения ключей. Их протокол распределения ключей, названный протоколом обмена ключами Диффи-Хеллмана, позволяет двум сторонам информационного обмена сформировать общий секретный ключ, используя открытый канал связи, и не прибегая к личной встрече. Стойкость этого протокола основывается на проблеме дискретного логарифмирования в конечной абелевой группе.

Рассмотрим протокол обмена ключами Диффи-Хеллмана для двух пользователей, которых традиционно обозначим A и B.

Открытыми параметрами протокола являются большое простое число p и образующий группы F_p^* g. Эти параметры постоянны и известны всем – как законным пользователям, так и возможному злоумышленнику.

Схема формирования пользователями общего секретного ключа приведена на рисунке.

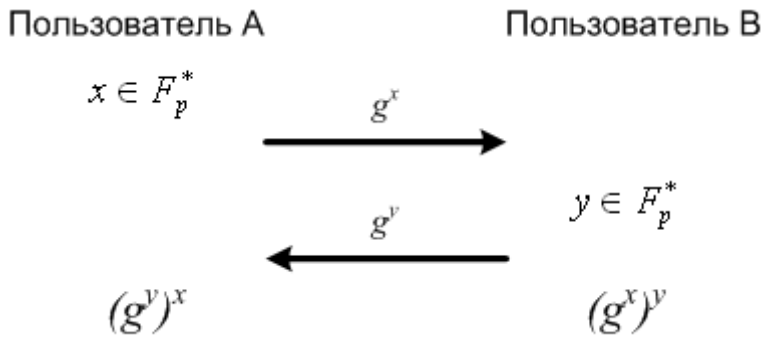


Рисунок 5 – Протокол Диффи-Хеллмана

Пользователь A генерирует большое число x, вычисляет $g^x \pmod p$ и отправляет полученное значение пользователю B. Пользователь B генерирует большое число y, вычисляет $g^y \pmod p$ и отправляет полученное значение пользователю A. Затем пользователь A вычисляет $\left(g^y\right)^x \pmod p$, а пользователь B вычисляет $\left(g^x\right)^y \pmod p$. Так как $\left(g^y\right)^x \pmod p = \left(g^x\right)^y \pmod p = g^{xy} \pmod p$, то пользователи A и B получают общий секретный ключ $k = g^{xy} \pmod p$.

Если пользоваться традиционной терминологией криптографии с открытым ключом, то значения x и y, которые необходимо держать в секрете, можно назвать закрытыми ключами пользователей A и B, а значения $g^x \pmod p$ и $g^y \pmod p$, соответственно, открытыми ключами.

Какие действия в данной ситуации может осуществить злоумышленник? Он может перехватить значения $g^x \pmod p$ и $g^y \pmod p$, которые пользователи A и B посылают друг другу. Но вычислить на основе перехваченных значений секретный ключ $g^{xy} \pmod p$ злоумышленник сможет, только определив x по известному $g^x \pmod p$ и возведя $g^y \pmod p$ в полученную степень. Таким образом, мы приходим к задаче дискретного

логарифмирования в конечном поле. Если значения p и g выбраны должным образом, такие вычисления практически невозможны.

Однако протокол обмена ключами Диффи-Хеллмана не защищен от атаки называемой «человек посередине», называемой также атакой посредника. В чем она заключается? Дело в том, что у законных участников информационного обмена не может быть твердой уверенности, что они общаются именно друг с другом. Возможна ситуация, изображенная на рисунке ниже.

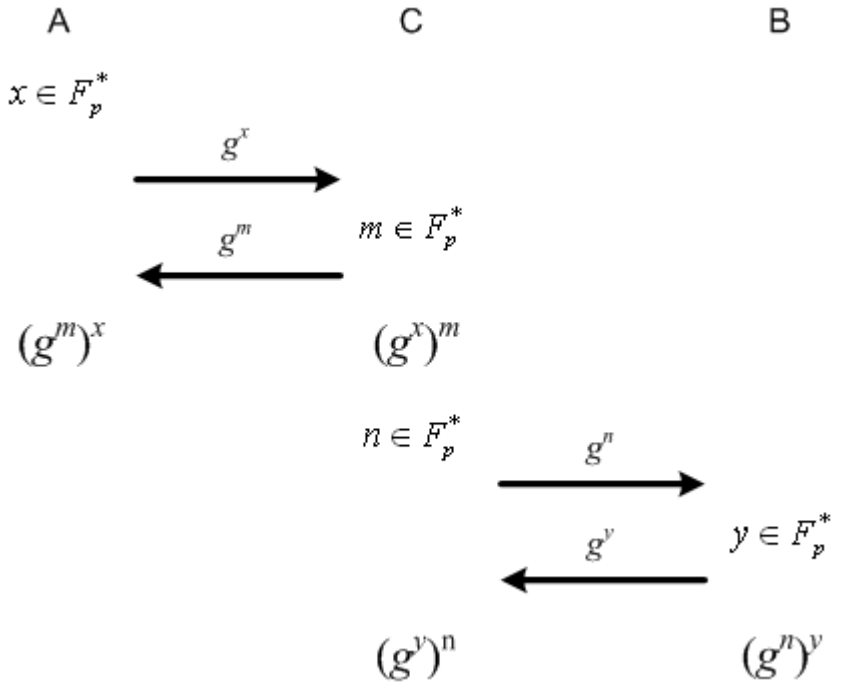


Рисунок 6 – Атака «человек посередине»

Как видно из рисунка, пользователь A договаривается об общем секретном ключе со злоумышленником C, думая, что договаривается с пользователем B. Пользователь B также договаривается об общем секретном ключе со злоумышленником, думая, что договаривается с пользователем A. В итоге вся переписка проходит через злоумышленника C, действия которого, если он не вносит изменений в сообщения, не могут быть обнаружены.

Один из способов пресечь атаку посредника на протокол обмена ключами Диффи-Хеллмана – это использование цифровой подписи. В этом случае стороны информационного обмена будут точно знать, с кем ведут переписку.

2. Криптосистема RSA

В 1978 году трое ученых, Рональд Райвест, Ади Шамир и Леонард Адлеман, опубликовали первый криптографический алгоритм с открытым ключом, названный ими (по первым буквам фамилий) RSA. Этот алгоритм основывается на сложности проблемы факторизации, то есть разложения числа на простые множители.

Рассмотрим информационный канал между двумя пользователями A и B, которых по традиции будем называть Алиса и Боб.

Алгоритм генерации ключей.

1. Алиса генерирует два больших простых числа p и q, отличных друг от друга. При этом $|p - q|$ – большое число, хотя p и q имеют приблизительно одинаковый битовый размер.

2. Держа p и q в секрете, Алиса вычисляет их произведение $n = p \cdot q$, которое называют модулем алгоритма.

3. Алиса вычисляет значение функции Эйлера для n по формуле $\varphi(n) = (p - 1)(q - 1)$.

4. Алиса выбирает целое число e, взаимно простое со значением функции $\varphi(n)$. Это число называется экспонентой шифрования. Как правило, e берут равным 3, 17 или 65537.

5. Алиса применяет расширенный алгоритм Евклида к паре чисел e и $\varphi(n)$ и вычисляет значение d, удовлетворяющее соотношению $e \cdot d \equiv 1 \pmod{\varphi(n)}$. Это значение называется экспонентой расшифрования.

6. Пара (e, n) публикуется в качестве открытого ключа Алисы, d является закрытым ключом и держится в секрете.

Алгоритм зашифрования.

1. Боб получает аутентичную копию открытого ключа Алисы – пару (e, n).

2. Боб представляет сообщение в виде числа m, меньшего модуля алгоритма. В общем случае сообщение может быть разбито на блоки, каждый из которых представляется своим числом.

3. Боб вычисляет $C = m^e \pmod{n}$.

4. Зашифрованное сообщение отправляется Алисе.

Алгоритм расшифрования.

1. Алиса получает криптограмму C от Боба.

2. Алиса вычисляет $m = C^d \pmod{n}$.

ПРАКТИЧЕСКОЕ ЗАНЯТИЕ № 7

Схемы аутентификации. Упрощенная схема идентификации Фейге-Фиата-Шамира. Схема идентификации Гиллу-Кискате. Протокол аутентификации Шнорра. Преобразование схем идентификации в схемы подписи. Схема подписи Фиата-Шамира. Схема подписи Гиллу-Кискате. Протокол цифровой подписи. Неоспоримая цифровая подпись Чаума (David Chaum). Подпись «вслепую». Подписи, подтверждаемые доверенным лицом.

Задание. Обмен информацией с использованием протокола Шамира

Краткое изложение основ и правил применения протокола Шамира с привязкой к практическим аспектам рассматриваются при проведении практического занятия на конкретном примере. Алгоритм и расчетные формулы подробно рассматриваются в разделе теоретического обучения (либо на лекционных занятиях, либо при выполнении СРС).

Пример. *Дано*: $p_A = 38177$, $p_B = 52631, M = "XYZ"$.

$e_A = 51407$ по условию

Этап 1. Генерация ключей

Абонент *A*: выберем значение $НОД(e_A, p-1) = 1$. Находим значение d_A:

$$d_A = e_A^{-1} (\text{mod } p_A - 1) = 51407^{-1} (\text{mod}(51407 - 1)) = 24335.$$

Абонент *B*: выберем значение $e_B = 42239$ по условию $НОД(e_B, p-1) = 1$. Находим значение d_B:

$$d_B = e_B^{-1} (\text{mod } p_B - 1) = 42239^{-1} (\text{mod}(52631 - 1)) = 32229.$$

Закрытый ключ абонента A: $e_A = 51407$.

Закрытый ключ абонента B: $e_B = 42239$.

Этап 2. Преобразование сообщения в числовой эквивалент

Преобразуем передаваемую триграмму “*XYZ*” в числовой эквивалент для последующей обработки

$$M = "XYZ" = 23 \cdot 26^2 + 24 \cdot 26^1 + 25 \cdot 26^0 = 16197.$$

Этап 3. Трехпроходный алгоритм Шамира

Используя обозначения, используемые в протоколе Шамира, введем значения α и β: $\alpha = e_A = 51407$; $\beta = e_B = 42239$.

Выполним 1-й *шаг* алгоритма Шамира:

$$C_1 = E_\alpha(M) = M^{e_A} (\text{mod } p_A) = 16197^{51407} (\text{mod } 38177) = 30944.$$

Выполним 2-й *шаг* алгоритма Шамира:

$$C_2 = E_\beta(C_1) = C_1^{\overset{e}{B}} (\mathrm{mod}\ p_B) = 30944^{\overset{42239}{\ }} (\mathrm{mod}\, 52631) = 34848.$$

Выполним 3-й *шаг* алгоритма Шамира:

$$C_3 = D_\alpha(C_2) = C_2^{\overset{d}{A}} (\mathrm{mod}\ p_A) = 34848^{\overset{24335}{\ }} (\mathrm{mod}\, 28177) = 14648.$$

Вычислим передаваемое абоненту *B* значение из выражения:

$$M = D_\beta(C_3) = D_\beta\left(E_\beta(M)\right):$$

$$M = D_\beta(C_3) = C_3^{\overset{d}{B}} (\mathrm{mod}\ p_B) = 14648^{\overset{34229}{\ }} (\mathrm{mod}\, 52631) = 16197.$$

Этап 4. Преобразование полученного значения к формату текста

$$M = 216197 = 3 \cdot 26^2 + 24 \cdot 26^1 + 25 \cdot 26^0 = "XYZ".$$

Полученное значение: *“XYZ”*.