1. **Цель лабораторной работы:**

Реализовать пороговую схему Асмута – Блума

Это пороговая схема разделения секрета, построенная с использованием простых чисел. Позволяет разделить секрет (число) между k сторонами таким образом, что его смогут восстановить любые n участников.

2. **Описание алгоритма и пояснения к заданиям:**

Схема Асмута – Блума - это пороговая схема разделения секрета, построенная с использованием простых чисел. Позволяет разделить секрет (число) между k сторонами таким образом, что его смогут восстановить любые n участников.

Пусть M – некоторый секрет, который нам требуется сохранить.

Выбирается простое число p, большее M. Выбирается n взаимно простых с друг другом чисел $d_1, d_2, d_3, \dots, d_n$, таких что:

- $\forall i : d_i > p$
- $\forall i : d_i < d_{i+1}$
- $d_1 * d_2 * \dots d_m > p * d_{n-m+2} * d_{n-m+3} * \dots * d_n$

Далее выбирается просто число $\boldsymbol{r}$ и вычисляется

$$M' = M + r * p$$

После этого вычисляются доли, которые потом раздаются участникам

$$k_i = M' \, mod \, d_i$$

$\{p, d_i, k_i\}$ – доли, которые раздаются участникам

Используя Китайскую Теорему об Остатках, можно восстановить секрет M, имея m и более долей.

При этом M' должно быть больше произведения долей неавторизованных участников.

3. Программная реализация:

Генерация параметров

```
int generateParams(int M, int n, vector<int>& d_num) {
    int p = generatePrime(M + 1, M + 5);

    d_num.push_back(generatePrime(p + 20, p + 40));

    for (size_t i = 0; i < n - 1; i++) {
        d_num.push_back(generatePrime(d_num[i] + 10, d_num[i] + 30));
    }

    return p;
}
```

Рис.1 – Функция генерации параметров

```
int generateM2(vector<int>& d_nums, int k, int M, int p) {
    int border = 1;
    size_t size = d_nums.size();

    for (size_t i = 0; i < k - 1; i++) {
        border *= d_nums[size - 1];
        size--;
    }

    int r = generateNum(border + 1, border + 10);

    int M2 = M + r * p;

    return M2;
}
```

Рис.2 – Функция генерации M'

Создание долей

```
void generateProportions(vector<vector<int>>& pr, vector<int>& d_nums, int k, int M, int p) {
    int M2;

    M2 = generateM2(d_nums, k, M, p);

    for (size_t i = 0; i < d_nums.size(); i++) {
        pr.at(i).push_back(p);
        pr.at(i).push_back(d_nums[i]);
        pr.at(i).push_back(M2 % d_nums[i]);
    }

    cout << M2 << endl;
}
```

Рис.3 – Функция создания долей

Восстановление секрета

```
int CRT(vector<vector<int>>& pr)
{
    int M = 1;
    int ans = 0;

    for (int i = 0; i < pr.size(); i++)
        M *= pr[i][1];
    for (int i = 0; i < pr.size(); i++)
    {
        int x, y;
        int Mi = M / pr[i][1];
        gcd(Mi, pr[i][1], x, y);
        ans = (ans + Mi * x * pr[i][2]) % M;
    }
    return (ans + M) % M;
}
```

Рис.4 – Реализация КТО

```
newPr.push_back(proportions[0]);
newPr.push_back(proportions[3]);
newPr.push_back(proportions[2]);
cout << newPr[0][1] << " " << newPr[0][2] << endl;
cout << newPr[1][1] << " " << newPr[1][2] << endl;
cout << newPr[2][1] << " " << newPr[2][2] << endl;

int result = CRT(newPr);

cout << result % p;
```

Рис.5 – Процесс восстановления секрета

```
int generateM2(v
void generatePro

int main() {
    int k = 3;
    int n = 4;
    int M = 5;
    int p;
```

```
Консоль отладки Microsoft Visua
First rule is true
Second rule is true
Third rule is true
332267
37 7
227 166
137 42
5
```

Рис.6 – Результат работы программы

4. Вывод:

При выполнении лабораторной работы была реализована пороговая схема разделения секрета Асмута – Блума.

Особенностью схемы (как и всех, используемых на практике) является то, что даже $k - 1$ сторон, собравшихся вместе, не смогут найти секрет даже методом полного перебора всех возможных вариантов.

5. **Литература:**
    1) А.Л. Чмора "Современная прикладная криптография"
    2) Б.Я. Рябко "Основы современной криптографии для специалистов в ИТ"
    3) Баричев, Серов “Основы современной криптографии”