

Программирование Аффинного шифра на языке программирования PHP

Стрельцова Марина Николаевна

Приамурский государственный университет им. Шолом-Алейхема

Студент

Аннотация

В данной статье описан процесс реализации Аффинного шифра на языке программирования PHP. Также рассмотрен принцип работы данного шифра и ограничения при шифровании информации данным способом.

Ключевые слова: PHP, шифр, подстановка, ключ.

Programming Affine Cipher in PHP Programming Language

Streltsova Marina Nikolaevna

Sholom-Aleichem Priamursky State University

Student

Abstract

This article describes the process of implementing an affine cipher in the PHP programming language. The principle of operation of this cipher, its weaknesses and limitations in encrypting information in this way are also considered.

Keywords: PHP, cipher, substitution, clue.

1. Введение

1.1 Актуальность исследования

Аффинный шифр — это частный случай более общего моноалфавитного шифра подстановки. К шифрам подстановки относятся также шифр Цезаря, ROT13 и Атбаш. Поскольку аффинный шифр легко дешифровать, он обладает слабыми криптографическими свойствами [1].

В рамках данной статьи рассмотрен принцип работы Аффинного шифра, ограничения при шифровании информации и его реализация на языке программирования PHP.

1.2 Обзор исследований

В статье В. А. Биктимерова описан процесс разработки информационной системы на языках программирования php, sql, html по проверке умений студентов определять классические криптоалгоритмы и находить ключи шифрования [2]. И. И. Баранкова и другие авторы разработали приложение на языке программирования Delphi для шифрования и дешифрования текстовых данных с помощью метода Цезаря [3]. В

исследовании А. А. Щукина и М. П. Митина была проведена работа по разработке программы, которая даёт возможность для удобной работы с RSA шифрами, позволяя сократить время шифровки и расшифровки текста [4]. А.С.Шульгина-Тарашук и другие авторы рассмотрели современные методы шифрования информации в своей статье [5]. В исследовании А. И. Макарова рассматривается один из простейших методов шифрования информации с помощью изображений, а также приведен алгоритм для шифрования путём наложения цветового водяного знака и для его дешифрования [6].

1.3 Цель исследования

Целью исследования является описание процесса реализации Аффинного шифра на языке программирования PHP.

2. Алгоритм шифрования и дешифрования

В Аффинном шифре каждой букве алфавита размера m ставится в соответствие число из диапазона $0 \dots m-1$. Затем при помощи модульной арифметики для каждого числа, соответствующего букве исходного алфавита, вычисляется новое число, которое заменит старое в шифротексте. Процесс шифрования можно описать следующей формулой:

$$E(x) = (Ax + B) \bmod m$$

где x - номер шифруемой буквы в алфавите; m - размер алфавита; A , B - ключ шифрования.

Для расшифровки вычисляется другая функция:

$$D(x) = A^{-1}(x - B) \bmod m$$

Где A^{-1} - число обратное A по модулю m . Обратное к A число существует только в том случае, когда A и m - взаимно простые. Значит, при отсутствии ограничений на выбор числа A , расшифрование может оказаться невозможным.

3. Результаты исследования

Для начала объявим переменную с алфавитом (строка 3) и напишем функцию «getInvMultKey» для проверки ключа A (Рис. 1).

```
1 <?php
2
3 $alphabet = "ABCDEFGHIJKLMNOPQRSTUVWXYZ";
4
5 for($i = 0; $i < strlen($alphabet); $i++){
6     $numAlph[$alphabet[$i]] = $i;
7 }
8
9 function getInvMultKey($multKey){
10     global $alphabet;
11
12     if($multKey < 0 || $multKey > strlen($alphabet)){
13         $multKey = $multKey % strlen($alphabet) + strlen($alphabet);
14     }
15     for($i = 0; $i < strlen($alphabet); $i++){
16         if($multKey * $i % strlen($alphabet) == 1){
17             $invMultKey = $i;
18             echo "Обратное число ключа A = " . $invMultKey . "\n";
19             return $invMultKey;
20         }
21     }
22     echo "У ключа A нет обратного числа, попробуйте другое значение\n";
23     return;
24 }
```

Рисунок 1 – Функция getInvMultKey

Если значение ключа A меньше нуля или больше длины алфавита, то вычисляем остаток числа A от деления на длину алфавита и прибавляем к нему длину алфавита (строки 9-14). Затем проверяем ключ A на наличие обратного числа с помощью цикла. Если данное число существует, то выводим соответствующее сообщение, а если отсутствует, то предупреждаем об этом пользователя.

Следующим шагом необходимо написать функцию «Encode» для кодирования сообщения (Рис. 2)

```
26 function Encode($txt, $addKey, $multKey){
27     global $alphabet, $numAlph;
28     $code = "";
29
30     $invMultKey = getInvMultKey($multKey);
31     if(!$invMultKey) return;
32
33     for($i = 0; $i < strlen($txt); $i++){
34         $code .= $alphabet[( $numAlph[$txt[$i]] * $multKey + $addKey) % strlen($alphabet)];
35     }
36     return $code;
37 }
38 }
```

Рисунок 2 – Функция Encode

Передаём функции сообщение (\$txt) и два ключа - А, В (\$multkey и \$addkey соответственно) (строка 26). Если ключа А не существует, то прерываем выполнение функции (строки 30-31). Далее с помощью ранее описанной формулы кодирования зашифровываем сообщение в цикле (строки 33-38).

Аналогичным образом напомним функцию «Decode», применяя формулу для расшифровки сообщения (Рис. 3).

```
40 function Decode($txt, $addKey, $multKey){
41     global $alphabet, $numAlph;
42     $code = "";
43
44     $invMultKey = getInvMultKey($multKey);
45     if(!$invMultKey) return;
46
47     for($i = 0; $i < strlen($txt); $i++){
48         $code .= $alphabet[((($numAlph[$txt[$i]] - $addKey) * $invMultKey) % strlen($alphabet))];
49     }
50     return $code;
51 }
```

Рисунок 3 – Функция Decode

Зашифруем и расшифруем несколько сообщений с различными ключами для проверки работоспособности кода, запустив его в терминале (Рис. 4-5).

```
56
57 echo Encode( txt: "POSTULAT", addKey: 2, multKey: 27) . "\n";
58 echo Decode( txt: "RQUVWNCV", addKey: 2, multKey: 27);
```

Terminal: Local × +

Обратное число ключа а = 1
RQUVWNCV
Обратное число ключа а = 1
POSTULAT

Рисунок 4 – Шифрование и дешифрование сообщения «POSTULAT»

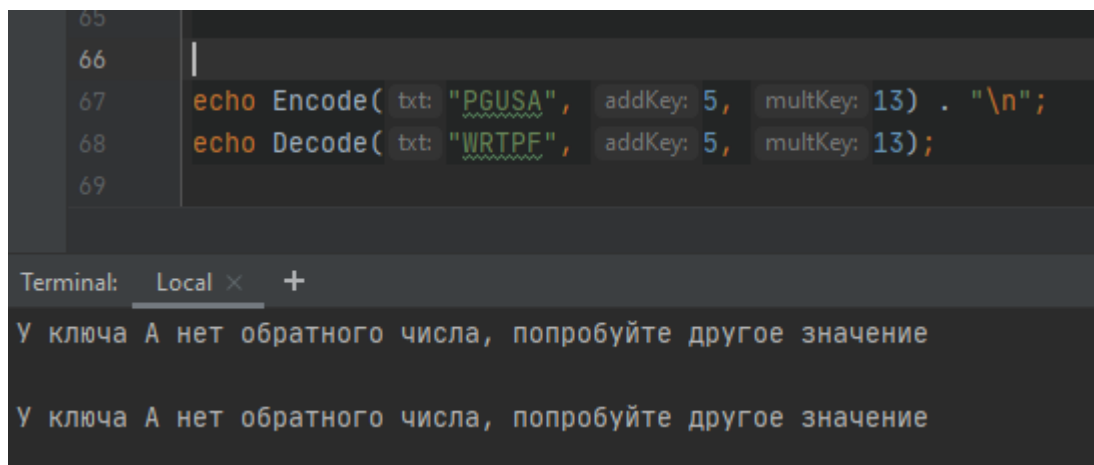
```
61
62 echo Encode( txt: "PGUSA", addKey: 6, multKey: 15) . "\n";
63 echo Decode( txt: "XSUQG", addKey: 6, multKey: 15);
64
```

Terminal: Local × +

Обратное число ключа А = 7
XSUQG
Обратное число ключа А = 7
PGUSA

Рисунок 5 – Шифрование и дешифрование сообщения «PGUSA»

Так как ключ A и m должны быть взаимно простыми для успешного шифрования и дешифрования сообщения, то можно проверить код, задав неверное значение (Рис. 6).



```
65
66
67 echo Encode( txt: "PGUSA", addKey: 5, multKey: 13) . "\n";
68 echo Decode( txt: "WRTPF", addKey: 5, multKey: 13);
69
```

Terminal: Local x +

У ключа A нет обратного числа, попробуйте другое значение

У ключа A нет обратного числа, попробуйте другое значение

Рисунок 6 – Проверка кода на ошибку

4. Выводы

В случае шифрования сообщений на русском языке (т.е. $m=33$) существует 297 нетривиальных Аффинных шифров, не учитывая 33 тривиальных шифра Цезаря. Это число легко посчитать, зная, что существует всего 20 чисел взаимно простых с 33 и меньших 33 (а это и есть возможные значения A) Каждому значению A могут соответствовать 33 разных дополнительных сдвига (значение B); то есть всего существует $20 \cdot 33$ или 660 возможных ключей. Аналогично, для сообщений на английском языке (то есть $m=26$) всего существует $12 \cdot 26$ или 312 возможных ключей. Такое ограниченное количество ключей приводит к тому, что система крайне некриптостойка с точки зрения принципа Керкгоффса [1].

Таким образом, в данной статье описан процесс реализации Аффинного шифра на языке программирования PHP. Также рассмотрен принцип работы данного шифра и ограничения при шифровании информации данным способом.

Библиографический список

1. Аффинный шифр URL: https://ru.wikipedia.org/wiki/Аффинный_шифр (дата обращения: 01.02.2023).
2. Биктимеров В. А. SaaS решение шифрования текста методом смещения, перестановки и гаммирования // ВІ-технологии и корпоративные информационные системы в оптимизации бизнес-процессов. 2018. С. 82-88.
3. Баранкова И. И. и др. Применение алгоритма шифрования методом Цезаря для шифрования данных, представленных в текстовом формате // Актуальные проблемы современной науки, техники и образования. 2014. Т. 2. С. 152-155.

4. Щукин А. А., Митин М. П. Программа для шифрования текста RSA-шифрованием. 2019.
5. Шульгина-Таращук А. С., Турдыбекова К. М., Турдыбекова К. К. Методы шифрования информации // Информационные технологии. 2019. С. 99-101.
6. Макаров А. И. Использование изображений для шифрования информации // Системы компьютерной математики и их приложения. 2017. №. 18. С. 91-93.