Лабораторная работа № 2

Тема: исследование и программирование классических методов симметричных криптосистем.

Цель: изучить теоретические основы шифров перестановки и замены, составить программу, позволяющую закодировать текст и выполнить обратное преобразование.

1 Теоретический вопрос

Rasterschlüssel 44 (RS44) — ручная техника шифрования, заменившая применявшийся прежде шифр «двойной квадрат. RS 44 использовался вооружёнными силами нацистской Германии, начиная с марта 1944 года

Имеется решетка из белых и черных квадратов, 24 строки на 25 колонок. В каждой строке в случайном порядке расположены 10 белых квадратов и 15 черных. Колонки и строки решетки обозначаются случайными и уникальными для каждой строки и колонки двухсимвольными обозначениями: aa, ab, ac, ad, ae, ba, bb и т.д., кроме того, колонки пронумерованы в случайном порядке.

Цифры и пунктуация записываются словами. Сообщение должно быть не короче 60 символов и не длиннее 200. Если оно длиннее, то его надо разбить на две шифровки. Если сообщение короче 60 символов, то до него и после надо добавить случайных слов. Чтобы отличить случайные слова от шифровки, надо повторить последние две буквы в «заголовке» и первые две в «хвосте».

Берем тонкий лист бумаги и накладываем на решетку. Черные квадраты просвечивают сквозь бумагу, так, что видно, где надо писать. Вначале надо выбрать стартовую колонку и строку, ее надо выбрать случайным образом, после чего зачеркнуть на решетке (не на верхнем листе бумаги) их обозначения одной чертой наискосок. Если она зачеркнута один раз, то можно второй раз по другой диагонали, если колонка или строка зачеркнута уже крестиком, то ее использовать в качестве начальной нельзя. Далее надо случайно выбрать колонку, с которой начнется шифровка, ее надо зачеркнуть крестиком, использовать ее потом в качестве начальной тоже нельзя.

В начале шифровки необходимо указать время шифрования, потом номер решетки, потом начальную строку и колонку, с которой начинается текст, затем колонку, с которой начинается шифровка (последнее — упрощенный вариант, действовавший с октября 44го, я описываю именно его). Для этого берется таблица кодирования начальных позиций и с ее помощью кодируются двухсимвольные обозначения колонок и строк.

Если мы начинаем писать открытый текст начиная с колонки bb и строки ae, то закодировать их данной таблицей можно, например так: tuzd (два разных символа выбрали из колонки «b», один из колонки «a», один из колонки «e». Шифровать по вертикали мы начнем с колонки ee, это можно закодировать, например как ny. Время 17:21, используется таблица номер 64: «1721-64-tuzd-ny». Это начало шифровки.

Далее все просто: текст заполняется в белых клетках слева направо, начиная с выбранной начальной позиции. Если текст доходит до конца листа, то продолжается с самого начала решетки сверху.

Затем списывается шифровка по вертикали, следуя нумерации колонок (если мы начали списывать шифр с колонки 23, то следующая колонка шифра 24, потом 25, потом 1,2 и т.д.).

Расшифровка происходит обратным образном: вначале определяется начальная клетка. Затем отсчитывается по белым квадратам количество символов в шифровке и отчеркивается все остальное неиспользованное сообщением пространство снизу и сверху, записывается по вертикали шифровка в соответствии с нумерацией колонок, а затем горизонтально читается сообщение.

2 Блок схемы

На рисунке 1 представлена блок схема шифрования сообщения.



Рисунок 1 – Блок схема шифрования

На рисунке 2 представлена блок схема дешифрования сообщения.



Рисунок 2 – Блок схема декодирования сообщения

3 Описание работы программы

На рисунке 3 представлен скриншот интерфейса разработанной программы.

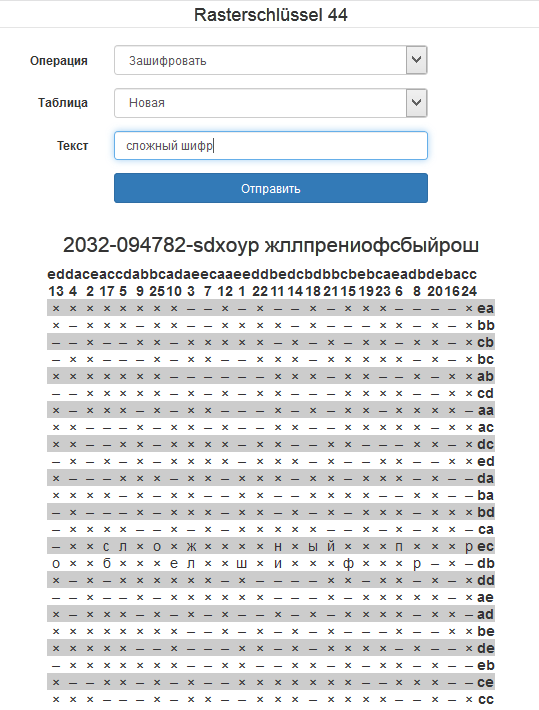


Рисунок 3 – Интерфейс программы

4 Текст программы

<?php

**class** Template {

**const** SIZE\_EMP = 10;

**const** SIZE\_COL = 25;

**const** SIZE\_ROW = 24;

**protected** $name;

**protected** $grid;

**protected** $table;

**protected** $names;

**protected** $numbers;

**protected** $used;

**public** **function** **\_\_construct**($name = **false**) {

**if**($name === **false** || $this->load($name) === **false**) {

$this->name = $name;

$this->generate();

$this->save();

}

}

**public** **function** generate() {

$this->names = [

['aa','ab','ac','ad','ae','ba','bb','bc','bd','be','ca','cb','cc','cd','ce','da','db','dc','dd','de','ea','eb','ec','ed','ee'],

['aa','ab','ac','ad','ae','ba','bb','bc','bd','be','ca','cb','cc','cd','ce','da','db','dc','dd','de','ea','eb','ec','ed']

];

$tmp = [];

$this->numbers = [];

$this->used = [[], []];

**for**($i=0;$i<self::SIZE\_COL;$i++) {

$tmp[] = $i < self::SIZE\_EMP ? '' : **false**;

$this->numbers[] = $i+1;

$this->used[0][] = 0;

}

**for**($i=0;$i<self::SIZE\_ROW;$i++) {

$this->used[1][] = 0;

}

**shuffle**($this->names[0]);

**shuffle**($this->names[1]);

**shuffle**($this->numbers);

$this->table = [];

**for**($i=0;$i<self::SIZE\_ROW;$i++) {

$chunk = $tmp;

**shuffle**($chunk);

$this->table[] = $chunk;

}

$grid = [];

**for**($i=0;$i<25;$i++) {

$grid[] = **chr**(**ord**('a')+$i);

}

**shuffle**($grid);

$this->grid = [

'a' => **array\_slice**($grid, 0, 5),

'b' => **array\_slice**($grid, 5, 5),

'c' => **array\_slice**($grid, 10, 5),

'd' => **array\_slice**($grid, 15, 5),

'e' => **array\_slice**($grid, 20, 5),

];

**return** $this;

}

**public** **function** load($name) {

**if**(**file\_exists**(\_\_DIR\_\_.'/tables/'.$name.'.json')) {

$data = **json\_decode**(**file\_get\_contents**(\_\_DIR\_\_.'/tables/'.$name.'.json'), **true**);

$this->name = $name;

$this->grid = $data['grid'];

$this->table = $data['table'];

$this->names = $data['names'];

$this->numbers = $data['numbers'];

$this->used = $data['used'];

**return** **true**;

}

**return** **false**;

}

**public** **function** save($name = **false**) {

$data = **json\_encode**([

'grid' => $this->grid,

'table' => $this->table,

'names' => $this->names,

'numbers' => $this->numbers,

'used' => $this->used,

]);

**if**($name === **false**) {

**if**($this->name === **false**) {

$this->name = **substr**(**sha1**($data), -6);

}

$name = $this->name;

}

**file\_put\_contents**(\_\_DIR\_\_.'/tables/'.$name.'.json', $data);

**return** **false**;

}

**public** **function** encrypt($text, &$table = **null**) {

$text = **str\_replace**(' ', "пробел", $text);

$position = $this->select\_position();

$start\_col = $position%self::SIZE\_COL;

$start\_row = $position/self::SIZE\_COL;

$column\_index = $this->numbers[**array\_rand**($this->numbers)];

$this->used[0][**array\_search**($column\_index, $this->numbers)] += 2;

$heading = **date**('Hi').'-'.$this->name.'-'.$this->grid\_replace($this->names[0][$start\_col].$this->names[1][$start\_row].$this->names[0][**array\_search**($column\_index, $this->numbers)]);

**for**($i=0;$i<**mb\_strlen**($text);$i++) {

**while**($this->table[$position/self::SIZE\_COL][$position%self::SIZE\_COL] === **false**) {

$position = ($position + 1)%(self::SIZE\_COL\*self::SIZE\_ROW);

}

$this->table[$position/self::SIZE\_COL][$position%self::SIZE\_COL] = **mb\_substr**($text, $i, 1);

$position = ($position + 1)%(self::SIZE\_COL\*self::SIZE\_ROW);

}

$table = $this->draw\_table();

$message = '';

**for**($i=0;$i<self::SIZE\_COL;$i++) {

$col = **array\_search**($column\_index, $this->numbers);

**for**($j=0;$j<self::SIZE\_ROW;$j++) {

**if**($this->table[$j][$col] !== **false**) {

$message .= $this->table[$j][$col];

}

}

$column\_index = $column\_index%self::SIZE\_COL + 1;

}

$this->cleanup();

$this->save();

**return** $heading.' '.$message;

}

**public** **static** **function** decrypt($text, &$table = **null**) {

**list**($heading, $text) = **explode**(' ', $text, 2);

$time = **substr**($heading, 0, 4);

$template = **substr**($heading, 5, 6);

$position = **substr**($heading, 12, 6);

$template = **new** Template($template);

**return** $template->decrypt\_raw($position, $text, $table);

}

**public** **function** decrypt\_raw($position, $text, &$table = **null**) {

$position = $this->grid\_replace($position, **true**);

$start\_row = **array\_search**(**substr**($position, 2, 2), $this->names[1]);

$start\_col = **array\_search**(**substr**($position, 0, 2), $this->names[0]);;

$column\_index = $this->numbers[**array\_search**(**substr**($position, 4, 2), $this->names[0])];

$len = **mb\_strlen**($text);

$start\_pos = $start\_row\*self::SIZE\_COL + $start\_col;

$position = $start\_pos;

**for**($i=0;$i<$len;$i++) {

**while**($this->table[$position/self::SIZE\_COL][$position%self::SIZE\_COL] === **false**) {

$position = ($position + 1)%(self::SIZE\_COL\*self::SIZE\_ROW);

}

$this->table[$position/self::SIZE\_COL][$position%self::SIZE\_COL] = PHP\_INT\_MIN;

$position = ($position + 1)%(self::SIZE\_COL\*self::SIZE\_ROW);

}

$k = 0;

**for**($i=0;$i<self::SIZE\_COL;$i++) {

$col = **array\_search**($column\_index, $this->numbers);

**for**($j=0;$j<self::SIZE\_ROW;$j++) {

**if**($this->table[$j][$col] === PHP\_INT\_MIN) {

$this->table[$j][$col] = **mb\_substr**($text, $k++, 1);

}

}

$column\_index = $column\_index%self::SIZE\_COL + 1;

}

$table = $this->draw\_table();

$message = '';

$position = $start\_pos;

**for**($i=0;$i<$len;$i++) {

**while**($this->table[$position/self::SIZE\_COL][$position%self::SIZE\_COL] === **false**) {

$position = ($position + 1)%(self::SIZE\_COL\*self::SIZE\_ROW);

}

$message .= $this->table[$position/self::SIZE\_COL][$position%self::SIZE\_COL];

$position = ($position + 1)%(self::SIZE\_COL\*self::SIZE\_ROW);

}

$this->cleanup();

**return** **str\_replace**('пробел', ' ', $message);

}

**private** **function** select\_position() {

**do** {

$i = **mt\_rand**(1, self::SIZE\_COL\*self::SIZE\_ROW);

} **while**($this->used[0][$i%self::SIZE\_COL] > 1 || $this->used[1][$i/self::SIZE\_COL] > 1);

**return** $i;

}

**private** **function** grid\_replace($names, $direction = **false**) {

$result = [];

**for**($i=0;$i<**strlen**($names);$i++) {

$char = **substr**($names, $i, 1);

**if**($direction) {

**foreach**($this->grid **as** $key => $list) {

**if**(**in\_array**($char, $list)) {

$c = $key;

**break**;

}

}

} **else** {

**while**(**in\_array**($c = $this->grid[$char][**array\_rand**($this->grid[$char])], $result));

}

$result[] = $c;

}

**return** **implode**('', $result);

}

**private** **function** cleanup() {

**for**($i = 0; $i<self::SIZE\_ROW;$i++) {

**for**($j=0;$j<self::SIZE\_COL;$j++) {

**if**($this->table[$i][$j] !== **false**) {

$this->table[$i][$j] = '';

}

}

}

}

**public** **function** draw\_table() {

**ob\_start**();

**printf**('<table>');

**printf**('<thead>');

**printf**('<tr>');

**foreach**($this->names[0] **as** $name) {

**printf**('<th>%s</th>', $name);

}

**printf**('<th></th>');

**printf**('</tr>');

**printf**('<tr>');

**foreach**($this->numbers **as** $i => $number) {

**printf**('<th class="%s">%s</th>', $this->draw\_used($this->used[0][$i]), $number);

}

**printf**('<th></th>');

**printf**('</tr>');

**printf**('</thead>');

**printf**('<tbody>');

**for**($i=0;$i<self::SIZE\_ROW;$i++) {

**printf**('<tr>');

**for** ($j=0; $j < self::SIZE\_COL; $j++) {

**printf**('<td>%s</td>', $this->draw\_value($this->table[$i][$j]));

}

**printf**('<th class="%s">%s</th>', $this->draw\_used($this->used[1][$i]), $this->names[1][$i]);

**printf**('</tr>');

}

**printf**('</tbody>');

**printf**('</table>');

**return** **ob\_get\_clean**();

}

**private** **function** draw\_value($value) {

**if**($value === **false**) {

**return** "&times;";

} **elseif**($value == '') {

**return** "–";

}

**return** $value;

}

**private** **function** draw\_used($used) {

**if**($used == 0) {

**return** '';

} **elseif**($used == 1) {

**return** 'half';

} **else** {

**return** 'full';

}

}

**public** **static** **function** available\_tables() {

$list = [];

**foreach**(**scandir**(\_\_DIR\_\_.'/tables') **as** $file) {

**if**(!**is\_dir**($file) && **preg\_match**('/\.json$/i', $file)) {

$list[] = **mb\_substr**($file, 0, **mb\_strlen**($file) - 5);

}

}

**return** $list;

}

}

Вывод

В ходе выполнения лабораторной работы были изучены теоретические основы шифров перестановки и замены, составлена программа, позволяющая закодировать текст, а также выполнить обратное преобразование. Был изучен шифр Rasterschlüssel 44.