Отчёт по лабораторной работе №7

Элементы криптографии. Однократное гаммирование

Косолапов Степан Эдуардович НПИбд-01-20

Содержание

Список иллюстраций

Список таблиц

1 Цель работы

Освоить на практике применение режима однократного гаммирования.

2 Выполнение работы

1. Код будем писать на языке Javascript. Напишем сперва вспомогательные фунцкии stringToHex и hexToString. Они будут переводить нам набор символов в шестнадцатеричные числа - соответствующие ASCII коду символа в кодировке UTF-16, разделённые пробелом:

```
const stringToHex = (str) => {
  return str.split('').map(c => c.charCodeAt(0).toString(16)).join(' ');
}

const hexToString = (hexStr) => {
  return hexStr.split(' ').map(c => String.fromCharCode(Number.parseInt(c, 16))).join('');
}
```

2. Напишем функцию, которая сгенерирует случайный набор символов(от 0 до 1048 в ASCII UTF-16) указанной длины, в шестнадцатеричном представлении:

```
const generateKey = (length) => {
  const result = [];

  for (let i = 0; i < length; i++) {
    const asciiCode = Math.floor(Math.random() * 1048);
    result.push(asciiCode.toString(16));
  }
```

```
  return result.join(' ');
}
```

3. Так же напишем основную функцию, которая и будет выполнять шифрование. Она принимает на вход два шестнадцатеричных набора(текст и ключ), и выполняет xor посимвольно, возвращая новый шестнадцатеричный набор:

```
const gammingCipher = (hexText, hexKey) => {
  const textSplit = hexText.split(' ');
  const keySplit = hexKey.split(' ');

  if (textSplit.length !== keySplit.length) {
    throw new Error('Key and message must have equal lengths.');
  }

  return textSplit.map((textCharHex, i) => {
    const keyCharHex = keySplit[i];

    const xorResult = Number.parseInt(textCharHex, 16) ^ Number.parseInt(keyCharHex, 16); // p_i xor k_i

    return xorResult.toString(16);
  }).join(' ')
}
```

4. Чтобы найти ключ, который расшифрует текст “Штирлиц – Вы Герой!”, как “С Новым Годом, друзья!” - нам нужно чтобы у этих текстов была одинаковая длина, поэтому допишем несколько восклицательных знаков к тексту “Штирлиц – Вы Герой!!!!”.

5. Теперь сгенерируем ключ для шифрования текста. Его длина(количество шестнадцатеричных чисел) должна быть 22, как у каждого из текстов:

```
const initialTextLength = 'Штирлиц – Вы Герой!!!!'.length;
```

```
console.log('initialTextLength:', initialTextLength);
console.log('generatedKey:', generateKey(initialTextLength));
```

Запустим программу:

```
node index.js
```

```
initialTextLength: 22
generatedKey: 203 3e7 2ea ec 2dc 29 3b4 10b 7f 23 33b 185 1ac 121 26f 97 1d5 3ad 1a3 97 25a 3c1
```

6. Имея ключ, зашифруем сообщение “Штирлиц – Вы Герой!!!!”:

```
const hexMessage = stringToHex(message);

console.log('charMessage:', message);
console.log('hexMessage:', hexMessage);

const encryptedMessage = gammingCipher(hexMessage, hexKey);

console.log('encryptedMessage:', encryptedMessage);
```

Вывод команды:

```
node index.js
```

```
charMessage: Штирлиц – Вы Герой!!!!
hexMessage: 428 442 438 440 43b 438 446 20 2013 20 412 44b 20 413 435 440 43e 439 21 21 21 21
hexKey: 203 3e7 2ea ec 2dc 29 3b4 10b 7f 23 33b 185 1ac 121 26f 97 1d5 3ad 1a3 97 25a 3c1
encryptedMessage: 62b 7a5 6d2 4ac 6e7 411 7f2 12b 206c 3 729 5ce 18c 532 65a 4d7 5eb 794 182 b6 27b 3e0
```

7. Теперь, чтобы найти ключ такой, который при дешифровке зашифрованного сообщения получал “С Новым Годом, друзья!” - нам достаточно сделать гаммирование текста “С Новым Годом, друзья!” зашифрованным сообщением. И мы получим нужный ключ.

```
const newYearMessage = 'С Новым Годом, друзья!';
const hexNewYearMessage = stringToHex(newYearMessage);

console.log('charNewYearMessage:', newYearMessage);
console.log('hexNewYearMessage:', hexNewYearMessage);

const hexNewYearKey = gammingCipher(encryptedMessage, hexNewYearMessage)

console.log('hexNewYearKey:', hexNewYearKey)
```

Вывод команды:

```
node index.js

charNewYearMessage: С Новым Годом, друзья!
hexNewYearMessage: 421 20 41d 43e 432 44b 43c 20 413 43e 434 43e 43c 2c 20 434 440 443 437 44c 44f 21
hexNewYearKey: 20a 785 2cf 92 2d5 5a 3ce 10b 247f 43d 31d 1f0 5b0 51e 67a e3 1ab 3d7 5b5 4fa 634 3c1
```

8. Проверим правильность решения:

```
console.log('decrypting initial message with hexNewYearKey...')

console.log('charDecrypted', hexToString(gammingCipher(encryptedMessage, hexNewYearKey)));
console.log('hexDecrypted', gammingCipher(encryptedMessage, hexNewYearKey));
```

Вывод команды:

```
node index.js

decrypting initial message with hexNewYearKey...
charDecrypted С Новым Годом, друзья!
hexDecrypted 421 20 41d 43e 432 44b 43c 20 413 43e 434 43e 43c 2c 20 434 440 443 437 44c 44f 21
```

3 Контрольные вопросы

1. Поясните смысл однократного гаммирования.

Однократное гаммирование - это способ шифрования, при котором каждый бит данных как-то комбинируется с отдельным, случайно сгенерированным ключом (или “гаммой”) той же длины.

2. Перечислите недостатки однократного гаммирования.

- Требуется генерация и безопасное хранение длинных случайных ключей той же длины, что и шифруемые сообщения.
- Ключи не могут быть переиспользованы, иначе это приведет к уязвимостям.
- Передача ключей между сторонами может быть трудоемким процессом.

3. Перечислите преимущества однократного гаммирования.

- При правильной реализации, она обеспечивает абсолютную криптографическую стойкость.
- Сам процесс шифрования и дешифрования обычно является простым и быстрым.
- Нет способа восстановить исходное сообщение без доступа к точному ключу.

4. Почему длина открытого текста должна совпадать с длиной ключа?

Если длина открытого текста не совпадает с длиной ключа, внедрение дополнительных символов или усечение ключа может привести к потере информации или обеспечивать недостаточную безопасность.

5. Какая операция используется в режиме однократного гаммирования, назовите её особенности?

В режиме однократного гаммирования используется операция XOR (исключающее “или”). Она обладает следующими особенностями: она обратима, коммутативна и ассоциативна; если дважды применить XOR с одним и тем же значением, исходное значение восстанавливается.

6. Как по открытому тексту и ключу получить шифротекст?

Шифротекст получается путем применения операции XOR к открытому тексту и ключу.

7. Как по открытому тексту и шифротексту получить ключ?

Ключ может быть получен путем применения операции XOR к шифротексту и открытому тексту.

8. В чем заключаются необходимые и достаточные условия абсолютной стойкости шифра?

Абсолютная стойкость шифра означает, что даже с бесконечными вычислительными ресурсами у атакующего не будет никаких шансов узнать информацию о исходном тексте, не зная ключ. Необходимыми и достаточными условиями для абсолютной стойкости шифра являются: использование действительно случайного ключа, который равен по длине, исходному сообщению, и использование ключа только один раз.

4 Выводы

В данной работе мы освоили на практике применение режима однократного гаммирования.