Учреждение образования

«БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНОЛОГИЧЕСКИЙ УНИВЕРСИТЕТ»

Исследование ассиметричных шифров

Студент: Коктыш Е. С.

ФИТ 3 курс 6 группа

Преподаватель: Нистюк О. А.

Минск 2024

1. **Теоретические сведения**

В основу асимметричной криптографии положена идея использовать ключи парами: один – для зашифрования (открытый или публичный ключ), другой – для расшифрования (тайный ключ). Отметим, что указанная пара ключей принадлежит получателю зашифрованного сообщения.

Все алгоритмы шифрования с открытым ключом основаны на использовании односторонних функций, к числу которых, как известно, относится вычисление дискретного логарифма.

Определение 1. Односторонней функцией (one-way function) называется математическая функция, которую относительно легко вычислить, но трудно найти по значению функции соответствующее значение аргумента, т. е., зная х, легко вычислить f(x), но по известному f(x) трудно найти подходящее значение x.

Алгоритмы шифрования с открытым ключом можно использовать для решения следующих задач:

* зашифрования/расшифрования передаваемых и хранимых данных в целях их защиты от несанкционированного доступа,
* формирования цифровой подписи под электронными документами,
* распределения секретных ключей, используемых далее при шифровании документов симметричными методами.

В данной работе мы будем работать над аспектами решения первой из указанных задач.

По мнению Диффи и Хеллмана алгоритм шифрования с открытым ключом, должен:

* вычислительно легко создавать пару (открытый ключ, e – закрытый ключ, d),
* вычислительно легко зашифровывать сообщение Mi открытым ключом,
* вычислительно легко расшифровывать сообщение Ci, используя закрытый ключ,
* обеспечивать непреодолимую вычислительную сложность определения соответствующего закрытого ключа при известном открытом ключе,
* обеспечивать непреодолимую вычислительную сложность восстановления исходного (открытого сообщения, Mi) зная только открытый ключ и зашифрованное сообщение, Ci.

Определение 2. Ранцевый (рюкзачный) вектор S = (s1, . . ., sn) – это упорядоченный набор из n, n ≥ 3, различных натуральных чисел si. Входом задачи о ранце (рюкзаке) называем пару (S, S), где S – рюкзачный вектор, а S – натуральное число.

Решением для входа (S, S) будет такое подмножество из S, сумма элементов которого равняется S.

В наиболее известном варианте задачи о ранце требуется выяснить, обладает или нет данный вход (S, S) решением. В варианте, используемом в криптографии, нужно для данного входа (S, S) построить решение, зная, что такое решение существует. Оба эти варианта являются NP-полными. Имеются также варианты этой задачи, которые не лежат даже в классе NP.

Как видим, проблема укладки ранца формулируется просто. Дано множество предметов общим числом n различного веса. Спрашивается, можно ли положить некоторые из этих предметов в ранец так, чтобы его вес стал равен определенному значению S? Более формально задача формулируется так: дан набор значений s1, s2, …, sn и суммарное значение S. Требуется вычислить значения si такие, что

S = b1\*s1 + b2\*s2+... + bn\*sn.

Здесь bi может быть либо нулем, либо единицей. Значение bi = 1 означает, что предмет mi кладут в рюкзак, а bi = 0 – не кладут.

Суть метода для шифрования состоит в том, что существуют две различные задачи укладки ранца: одна из них решается легко и характеризуется линейным ростом трудоемкости, а другая решается трудно. Легкий для укладки ранец можно трансформировать в трудный.

Трудный для укладки ранец применяется в качестве открытого ключа е, который легко использовать для зашифрования, но невозможно – для расшифрования. В качестве закрытого ключа d применяется легкий для укладки ранец, который предоставляет простой способ расшифрования сообщения.

В качестве закрытого ключа d (легкого для укладки ранца) используется сверхвозрастающая последовательность, состоящая из n элементов: d1, d2, …, dz: d = {di}, i = 1, …, n.

Определение 3. Сверхвозрастающей называется последовательность, в которой каждый последующий член больше суммы всех предыдущих.

Необходимо по очереди анализировать некоторый «текущий вес» S предметов, составляющих сверхвозрастающую последовательность; в результате анализа нужно упаковать (доупаковать) ранец.

1. В качестве текущего выбирается число S, которое сравнивается с «весом» самого тяжелого предмета (dn); если текущий вес меньше веса данного предмета, то его в ранец не кладут (0), в противном случае его укладывают (bn = 1) в ранец и переходят к анализу очередного (в общем случае – i-го предмета).

2. Если на предыдущем (i-м шаге) предмет пополнил ранец, то текущий вес уменьшают на вес положенного предмета (S = S – di); переходят к следующему по весу предмету в последовательности: di-1.

Шаги повторяются до тех пор, пока процесс не закончится.

Если текущий вес уменьшится до нуля (S = 0), то решение найдено. В противном случае – нет

Открытый ключ e представляет собой нормальную (не сверхвозрастающую) последовательность. Он формируется на основе закрытого ключа и не позволяет легко решить задачу об укладке ранца.

Для получения открытого ключа e (e = {ei}, i = 1, …, n) все значения закрытого ключа умножаются на некоторое число a по модулю n:

ei = di \* a (mod n).

Значение модуля n должно быть больше суммы всех чисел последовательности; кроме того, НОД (а, n) = 1.

Для расшифрования сообщения получатель (использует свой тайный ключ, d: сверхвозрастающую последовательность) должен сначала определить обратное к а число: а-1, такое что:

а \* а-1 (mod n) = 1.

Для вычисления обратных чисел по модулю можно использовать известный нам расширенный алгоритм Евклида.

После определения обратного числа каждое значение шифрограммы (ci) преобразуется в соответствии со следующим соотношением:

Si = ci \* а-1 mod n

Полученное на основании последней формулы для каждого блока число далее рассматривается как заданный вес ранца, который следует упаковать по изложенному выше алгоритму, используя сверхвозрастающую последовательность (тайный ключ получателя).

С другой стороны, известный способ определения, какие предметы кладутся в ранец, является проверка возможных решений до получения правильного. Самый быстрый алгоритм, принимая во внимание различную эвристику, имеет экспоненциальную зависимость от числа возможных предметов. Если добавить к последовательности весов еще один член, то найти решение станет вдвое труднее. Это намного труднее сверхвозрастающего ранца, где, при добавлении к последовательности одного элемента, поиск решения увеличивается на одну операцию.

Ранцевые криптосистемы не являются криптостойкими. А. Шамир и Р. Циппел обнаружили, что, зная числа а, a-1 и n («секретную лазейку»), можно восстановить сверхвозрастающую последовательность по нормальной последовательности. Важно то, что числа а и n («секретная пара») не обязательно должны быть теми же, что использовались при создании системы легальным пользователем.

1. **Практическая часть**

В данной лабораторной работе необходимо разработать авторское оконное приложение, которое должно реализовывать следующие операции:

• генерация сверхвозрастающей последовательности (тайного ключа);

• вычисление нормальной последовательности (открытого ключа);

• зашифрование сообщения, состоящего из собственных фамилии, имени и отчества;

• расшифрование сообщения;

• оценка времени выполнения операций зашифрования и расшифрования.

**2.1. Зашифрование**

После ввода текста, который необходимо зашифровать, приложение начинает свою работу, генерируя сверхвозрастающую последовательность (тайный ключ), нормальную последовательность (открытый ключ), затем сообщение шифруются и расшифровывается, вместе с тем вычисляется время работы программы.

Функция принимает на вход начальное число последовательности и длину генерируемой последовательности. Начальное число последовательности – это случайное 100-битное число. Функции для генерации тайного ключа, представленна в листинге 2.1.

|  |
| --- |
| var genPrivateKey = function (initialNumber, z) {      var sequence = [];      var element = initialNumber;      var sum = initialNumber;      for (var i = 0; i < z; i++) {          sequence.push(element);          element = sum.add((0, big\_integer\_1.default)(z));          sum = sum.add(element);      }      return sequence;  }; |

Листинг 2.1 – Функция генерации тайного ключа

Сверхвозрастающей называется последовательность, в которой каждый последующий член больше суммы всех предыдущих. Для генерации сверхвозрастающей последовательности реализована следующая функция, отображённая в листинге 2.2.

|  |
| --- |
| var getND = function (number, modulus) {      var m0 = modulus;      var y = big\_integer\_1.default.zero;      var x = big\_integer\_1.default.one;      if (modulus.eq(1)) {          return big\_integer\_1.default.zero;      }      while (number.gt(1)) {          var quotient = number.divmod(modulus).quotient;          var temp = modulus;          modulus = number.divmod(modulus).remainder;          number = temp;          temp = y;          y = x.minus(quotient.times(y));          x = temp;      }      if (x.lt(0)) {          x = x.plus(m0);      }      return x;}; |

Листинг 2.2 – Код генерации сверхвозрастающей последовательности

Далее, на основе сгенерированного приватного ключа подбираются параметры n и a для публичного ключа. Значение модуля n должно быть больше суммы всех чисел последовательности. Функция принимает приватный ключ и параметры a и n и генерирует публичный ключ. Код функции представлен в листинге 2.3.

|  |
| --- |
| var genPK = function (privateKey, a, n) {      var sequence = [];      var d;      var e;      for (var i = 0; i < privateKey.length; i++) {          d = privateKey[i];          e = d.multiply(a).mod(n);          sequence.push(e);      }      return sequence;  }; |

Рисунок 2.3 – Функция генерации публичного ключа

Результат генерации публичного ключа для сообщения в кодах ASCI и base64 представлен на рисунке 2.1. и 2.2.

|  |
| --- |
|  |

Рисунок 2.1 – Результат генерации публичного ключа для сообщения(ASCI)

|  |
| --- |
|  |

Рисунок 2.2 – Результат генерации публичного ключа для сообщения(base64)

После чего происходит шифрование сообщения. Для этого оно разбивается на блоки, по размеру равным размеру тайного ключа. В нашем случае 8. Затем, считая, что 1 указывает на присутствие элемента последовательности в ранце, а 0 – на его отсутствие, вычисляются полные веса рюкзаков.  Код функции представлен в листинге 2.4.

|  |
| --- |
| var decrypt = function (privateKey, encryptedText, a, n) {      var decryptedBytes = [];      var binaryResult = "";      var inverse = (0, mathUtils\_1.getInverseNumber)(a, n);      for (var \_i = 0, encryptedText\_1 = encryptedText; \_i < encryptedText\_1.length; \_i++) {          var cipher = encryptedText\_1[\_i];          var decryptedValue = cipher.times(inverse).mod(n);          var binaryString = (0, exports.getDecryptedBinary)(decryptedValue, privateKey);          binaryResult += binaryString;          var decryptedByte = parseInt(binaryString, 2);          decryptedBytes.push(decryptedByte);      }      return { decoded: new Uint8Array(decryptedBytes), binary: binaryResult };  }; |

Листинг 2.4 – Код функции шифрования

Результат шифрования текста (в кодах ASCII) представлен на рисунке 2.3.

|  |
| --- |
|  |

Рисунок 2.3 – Шифрование текста, представленного в кодах ASCII

Результат шифрования текста (в кодах base64) представлен на рисунке 2.4.

|  |
| --- |
|  |

Рисунок 2.4 – Шифрование текста, представленного в кодах base64

# 2.2. Расшифрование

Для расшифрования сообщения получатель (используя свой тайный ключ ***d***: сверхвозрастающую последовательность) должен сначала определить такое обратное к *а* число *а–*. После определения обратного числа каждое значение шифрограммы (*ci*) преобразуется в соответствии со следующим соотношением:

https://lh7-us.googleusercontent.com/SSUkUPm5kNJnXKjaXPiP-VnzBfwygEMTa2il7yHfzsrUMtS-iC9i-TU-zxtDxOluS6utI2qO9zaMs50beQecg79YHZlFiDoiMy7s8hb14bHGOfa0fCst5rIBF_yEKN2ZDaZfS6GxIBnOaa8Rd0ZQHA

Полученное на основании последней формулы для каждого блока число далее рассматривается как заданный вес ранца, который следует упаковать, используя сверхвозрастающую последовательность, то есть тайный ключ получателя.

Код функции расшифрования представлен в листинге 2.5.

|  |
| --- |
| var encrypt = function (publicKey, plaintext, encoding) {      var encryptedList = [];      if (encoding === Encoding.BASE64) {          plaintext = (0, base64\_1.base64Encode)(plaintext);      }      plaintext.split('').forEach(function (b, index) {          var binaryString;          if (encoding === Encoding.ASCII) {        binaryString = plaintext.charCodeAt(index).toString(2).padStart(8, '0');          }          else {              binaryString = (0, base64\_1.convertBase64ToBinary)(plaintext[index]);          }          var positions = [];          for (var i = 0; i < binaryString.length; i++) {              if (binaryString[i] === '1') {                  positions.push(i);              }          }          var sum = big\_integer\_1.default.zero;          positions.forEach(function (position) {              if (position < publicKey.length) {                  sum = sum.add(publicKey[position]);              }          });          encryptedList.push(sum);      });      return encryptedList; |

Листинг 2.5 – Код функции расшифрования

Результат расшифрования текста (в кодах ASCII) представлен на рисунке 2.4.

|  |
| --- |
|  |

Рисунок 2.4 – Расшифрование текста, представленного в кодах ASCII

Результат шифрования текста (в кодах base64) представлен на рисунке 2.5.

|  |
| --- |
|  |

Рисунок 2.5 – Засшифрование текста, представленного в кодах base64

# 3. Время выполнения

Для оценки времени выполнения операций зашифрования и расшифрования была проведена оценка с разными таблицами кодировки, а также при подаче на вход алгоритма генерации ключей разных чисел z, которые отвечают за количество членов в ключевой последовательности.

Оценку времени операций зашифрования при использовании разных таблиц кодировки можно увидеть на рисунке 3. Разница между ними несущественная, однако, Base64 имеет тенденцию к немного более медленному зашифрованию и расшифрованию.

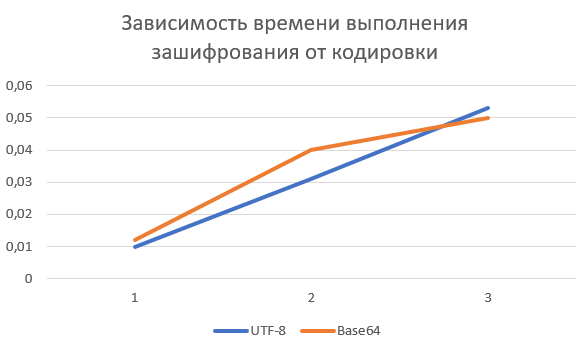


Рисунок 3 — График зависимости времени выполнения операции зашифрования от кодировки

График, отображающий изменение времени зашифрования при разном количестве членов ключевой последовательности.

Анализируя этот график, можно заметить, что при увеличении числа членов ключевой последовательности, время зашифрования значительно увеличивается. Однако, необходимо учитывать, что если будет использоваться длинная ключевая последовательность и короткое сообщение, то влияние параметра *z* можно и не ощутить.

# Вывод

В ходе выполнения данной лабораторной работы мы изучили и приобрели практические навыки разработки и использования приложений для реализации ассиметричных шифров. При увеличении числа членов ключевой последовательности увеличивается и время выполнения операций зашифрования/расшифрования. При использовании кодировок UTF-8 и Base64 время выполнения операция отличается незначительно.