**1. Введение**

Вопрос защиты данных при передаче по каналам связи уже давно имеет очень высокое значение во многих сферах жизни – личной переписке, передаче корпоративных данных, секретных документов и многого другого. Самым простым способом защитить данные от перехвата была бы организация защищенного канала связи. Однако на практике эта задача очень сложна в исполнении, а зачастую попросту невозможно. Поэтому для защиты данных применяется шифрование.

На данный момент существует большое количество видов шифрования, однако все их можно разделить на 3 вида – симметричное, асимметричное и гибридное. Однако до сих пор не существует абсолютно надежного способа защиты информации. Симметричное и асимметричное шифрование основано на использовании ключа – для шифрования и расшифровки. В симметричном шифровании используется один и тот же ключ, а в асимметричном – разные для этих операций. Наука о методах расшифровки зашифрованной информации без предназначенного для такой расшифровки ключа называется криптоанализом. Попытку раскрытия конкретного шифра с применением методов криптоанализа называют криптографической атакой на этот шифр.

Цель работы – рассмотреть некоторые способы криптоанализа асииметричного алгоритма шифрования RSA. Это криптографический алгоритм с открытым ключом, основывающийся на вычислительной сложности задачи факторизации больших целых чисел. Криптосистема RSA стала первой системой, пригодной и для шифрования, и для цифровой подписи. В качестве методов атаки рассмотрены следующие алгоритмы: метод дискретного логарифмирования Шенкса (метод больших и малых шагов), метод факторизации ро-Полларда и метод факторизации Ферма.

**2. Атака с помощью факторизации методом Ферма**

Метод факторизации Ферма — алгоритм факторизации (разложения на множители) нечётного целого числа n, предложенный Пьером Ферма [2]. Метод основан на поиске таких целых чисел x и y, которые удовлетворяют соотношению что ведёт к разложению .

Для разложения на множители нечётного числа n ищется пара чисел (x,y) таких, что , или . При этом числа (x+y) и (x-y) являются множителями n, возможно, тривиальными (то есть одно из них равно 1, а другое — n.)

Для разложения на множители нечётного числа n ищется пара чисел (x,y) таких, что x^2-y^2=n, или (x-y)\cdot (x+y) = n. При этом числа (x+y) и (x-y) являются множителями n, возможно, тривиальными (то есть одно из них равно 1, а другое — n.)

В нетривиальном случае, равенство x^2-y^2=n равносильно x^2-n=y^2, то есть тому, что x^2-n является квадратом.

Поиск квадрата такого вида начинается с   — наименьшего числа, при котором разность x^2-n неотрицательна.

Для каждого значения k \in \mathbb{N}, начиная с k=1, вычисляют   и проверяют, не является ли это число точным квадратом. Если не является, то k увеличивают на единицу и переходят на следующую итерацию.

Если  является точным квадратом, то есть   то получено разложение:

 n = x^2-y^2 = (x+y)(x-y) = a \cdot b, в котором

Если оно является тривиальным и единственным, то n — простое.

Проведя факторизацию n из открытого ключа, можно легко вычислить функцию , и после этого – закрытый ключ *d.*

**3. Атака с помощью факторизации ро-алгоритмом Полларда**

Ро-алгоритм (\rho-алгоритм)  – предложенный Джоном Поллардом в 1975 году алгоритм, служащий для факторизации (разложения на множители) целых чисел. Алгоритм наиболее эффективен при факторизации составных чисел с достаточно малыми множителями в разложении. ρ-алгоритм Полларда строит числовую последовательность, элементы которой образуют цикл, начиная с некоторого номера *n*, что может быть проиллюстрировано, расположением чисел в виде греческой буквы ρ, что послужило названием семейству алгоритмов.

Пусть N составное целое положительное число, которое требуется разложить на множители. Алгоритм выглядит следующим образом[3]:

Шаг 1. Случайным образом выбирается небольшое число x_{0} и строится последовательность \{x_{n}\}, n = 0, 1, 2, ..., определяя каждое следующее как x_{n+1} = F(x_{n})\, (\mathrm{mod}\,\, N).

Шаг 2. Одновременно на каждом *i*-ом шаге вычисляется   для каких-либо i, j таких, что j<i, например, i = 2j.

Шаг 3. Если d>1, то вычисление заканчивается, и найденное на предыдущем шаге число d является делителем N. Если N/d не является простым числом, то процедуру поиска делителей продолжается, взяв в качестве N число N'=N/d.

На практике функция F(x) выбирается не слишком сложной для вычисления (но в то же время не линейным многочленом), при условии того, что она не должна порождать взаимно однозначное отображение. Обычно в качестве F(x) выбираются функции F(x) = x^2 \pm 1 (\mathrm{mod}\, N)или F(x) = x^2 \pm a (\mathrm{mod}\, N). Однако функции x^2-2 и x^2 не подходят.

Если известно, что для делителя p числа N справедливо p \equiv 1\, (\mathrm{mod}\, k) при некотором k > 2, то имеет смысл использовать F(x) = x^k + b.

Аналогично с предыдущим методом, проведя факторизацию n из открытого ключа, можно легко вычислить функцию , и после этого – закрытый ключ *d.*

**4. Атака методом Шенкса для дискретного логарифмирования (больших и малых шагов)**

Алгоритм Гельфонда  – Шенкса (также называемый алгоритмом больших и малых шагов) — в  дискретного логарифмирования в мульпликативной группе кольца вычетов по модулю. Метод теоретически упрощает решение задачи дискретного логарифмирования, на вычислительной сложности которой построены многие криптосистемы с открытым ключом. Относится к методам встречи посередине.

Алгоритм 4.2.1 (алгоритм малых и больших шагов). Алгоритм  
вычисляет дискретный логарифм от у по основанию а и по модулю n,  
такой что у = ах (mod n)[3].

Шаг 1. Инициализация. Вычислить

Шаг 2. Вычисление маленького шага. Вычислить первую последователь-  
ность (список) S, состоящую из пар , r = 0,1,2,3,..., s - 1: S = и отсортировать S по *уаr* — первому элементу пар в S.

Шаг 3. Вычисление большого шага. Вычислить вторую последователь-  
ность (список) Т, состоящую из пар (*ats,ts*), t = 0,1,2,3,..., s: T = и отсортировать Т по *ats* — первому элементу пар в Т.

Поиск, сравнение и вычисление. Найти в обоих списках соответ-  
ствие *уаr = ats*, где *уаr* из S, a *ats* из Т, затем вычислить *х = ts - rх*  
и будет требуемым значением *loga у (*mod *n).*

**5. Вычислительные эксперименты**

Помимо реализации вышеуказанных методов было проведено сравнение их быстродействия, которое указано в табл. 1. Полученные данные показывают, что при n ~ 6\*108 работа всех методов идёт меньше секунды, но видно, что метод дискретного логарифмирования работает дольше других. При этом скорость его работы зависит от *e*. С ростом порядка n тенденция сохраняется, однако помимо этого проявляется различие в скорости работы методов факторизации. Так как метод ро-Полларда носит вероятностный характер, то в таблицу заносится среднее арифметическое 10 результатов эксперимента.

Табл. 1. Сравнение времени работы алгоритмов

|  |  |  |  |
| --- | --- | --- | --- |
|  | Время работы, секунд | | |
| n | Дискретное логарифмирование | Факторизация методом Ферма | Факторизация методом ро-Полларда |
| ≈6\*108 | 0.81 | <0.01 | <0.01 |
| ≈1.5\*1010 | 1.9 | 0.03 | 0.01 |
| ≈2.8\*1015 | >120 | 44 | 0.5 |

Из полученных данных можно сделать вывод, что при любых параметрах в скорости работы метод дискретного логарифмирования проигрывает. С ростом значения n актуальным становится только метод ро-Полларда, так как при значениях n ≈ 2.8\*1015 только этот метод даёт хорошие значения времени выполнения.