# Описание побочных атак на РША

## 2.1 Атака на малую открытую экспоненту

Первая побочная атака, называемая также атакой Хастада [], связана с выбором малой величины экспоненты шифрования *e­* в целях ускорения операции шифрования. Например, *e*=3 часто используется на практике, поскольку шифрование в таком случае требует выполнения одной операции возведения в квадрат и одной операции умножения*.*

Малая открытая экспонента может быть использована злоумышленником, когда одно и то же сообщение транслируется нескольким участникам. Допустим, легитимный пользователь хочет отправить *k* получателям одинаковое сообщение *M*, при этом для увеличение скорости шифрования криптосистема спроектирована таким образом, что используется одинаковая малая шифрующая экспонента *e* *≤ k*, но разные модули *ni*, а шифруемое сообщение *M* меньше любого из модулей *ni*. Тогда, имея любые *e* криптограмм из *С1*, …, *Сk* и *e* соответствующих наборов открытых ключей из (*e*, *n1*), …, (*e*, *nk*), можно восстановить исходное сообщение *M* в полиномиальное время. Совокупность криптограмм и открытых ключей можно представить в виде системы уравнений:

|  |  |
| --- | --- |
|  | (1) |

При этом, как уже отмечалось, для выполнения атаки злоумышленнику достаточно иметь систему из любых *e* уравнений системы (1). Весьма вероятно, что будет также выполнено условие:

gcd(*ni*, *nj*) = 1 при *i* ≠ *j*.

Тогда, согласно китайской теореме об остатках [**Ошибка! Источник ссылки не найден.**], существует общее решение системы (1):

,

которое может быть найдено по следующей формуле:

|  |  |
| --- | --- |
| , | (2) |

где ,  и .

Известно, что сообщение *M* < *ni* для *i = 1,2, …, k*, поэтому , а это значит, что исходное сообщение *M* может быть вычислено любым злоумышленником, который знает только криптограммы и не знает факторизацию модуля n или секретную экспоненту d, путем нахождения корня степени *e* по решению (2) системы (1):

.

Реализуя на практике данную атаку, для нахождения корня степени *e* можно использовать итерационный метод Ньютона [ ], представляющий из себя последовательность приближений и обладающий квадратичной скоростью сходимости [ ]. Приближение корня на каждом следующем шаге  вычисляется с использованием приближения на предыдущем шаге  по формуле:

|  |  |
| --- | --- |
| , | (3) |

где *e* – показатель корня,  – число, из которого извлекается корень [**.**].

Как сказано в [**.**], c целью уменьшения числа итераций при вычислении корней с использованием метода Ньютона в качестве начальной оценки корня удобно выбрать:

|  |  |
| --- | --- |
| , | (4) |

где – битовая длина числа.

Поскольку при выполнении атаки на малую открытую экспоненту заранее известно, что корень, который необходимо найти, является целым числом, результат вычисления на каждой итерации (3) округляется до ближайшего целого. В итоге, так как метод является сходящимся, на определенном этапе итерации полученное значение *ai+1* совпадет со значением *ai­­­*, полученным на предыдущем этапе, что и будет являться целочисленным корнем степени *e* из .

***Метод Ньютона***

Метод Ньютона – это итерационный численный метод для нахождения корней уравнения, основанный на принципах метода простой итерации [ ].

В свою очередь идея метода простой итерации состоит в том, чтобы уравнение вида:

привести к эквивалентному уравнению

так, чтобы отображение было сжимающим (сжимающее отображение - это отображение метрического пространства в себя, уменьшающее расстояние между любыми двумя точками не менее чем в 1 раз). Если это удается, то последовательность итераций сходится.

Такое преобразование можно делать разными способами. В частности, сохраняет корни уравнение вида:

если на исследуемом отрезке. В качестве можно выбрать:

,

что превращает метод простых итераций в «метод Ньютона». С учетом этого функция определяется как:

При условии, что функция дважды непрерывно-дифференцируемая в своей области определения и ее производная нигде не обращается в нуль, функция в окрестности корня осуществляет сжимающее отображение.

Арифметическим корнем *n*-ой степени положительного действительного числа A называется положительное действительное решение уравнения . Тогда задача нахождения корня n-ной степени может быть рассмотрена, как задача нахождение решения уравнения:

|  |  |
| --- | --- |
|  | (5) |

Производная этой функции равна . Тогда итерационное правило будет иметь вид:

,

что совпадает с (3).

Геометрически итерационный процесс метода Ньютона означает замену *k*-той итерации графика функции *y=f(x)* на касательную к этой функции в точке (рис. 2.1). В связи с этим метод иногда называют методом касательных. Уравнение касательной имеет вид

Найдем точку пересечения с осью Ox этой касательной, что соответствует нахождению решения линейного уравнения:

вместо нелинейного уравнения (5).

Выражая *x*, получаем:

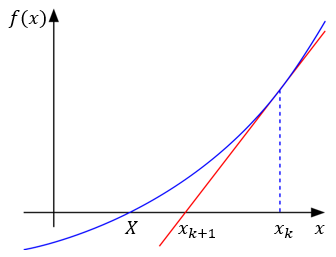


Рис. 2.12.1 Иллюстрация метода Ньютона

Теперь покажем, как итерационная формула может быть получена аналитически. Рассмотрим уравнение , где *X* – его корень, – *k*-ое приближение к корню. Тогда по теореме Лагранжа о средних значениях имеем:

,

где . Заменяя на значение (то есть используя предыдущее приближение к корню), приходим к приближенному равенству

.

Откуда получаем .

***Численный пример выполнения атаки***

Рассмотрим простой пример. Допустим, выбрана одинаковая малая шифрующая экспонента *e1*= *e2*= *e3*=3.

Модули криптосистемы равны:



Допустим, легитимный пользователь зашифровывает одинаковое сообщение *M*=83 и отправляет три различные криптограммы трем пользователям:



Если злоумышленник перехватил эти криптограммы, то для вычисления исходного сообщения ему необходимо решить систему уравнений, используя китайскую теорему об остатках (2), а затем вычислить кубический корень из решения этой системы.

Используя перехваченные криптограммы и пару чисел (*e*, *n*) открытого ключа, злоумышленник составляет систему уравнений:

|  |  |
| --- | --- |
|  | (6) |

Затем вычисляет  и :



и соответствующие обратные элементы:



Решение системы (6) вычисляется, как:



Теперь найдем корень третьей степени из 571787, используя метод Ньютона. В качестве начального приближения воспользуемся формулой (4). Вычислим начальное приближение по формуле (4): , где 20 – битовая длина числа 571787, а 3 – показатель корня.

|  |  |
| --- | --- |
|  | (7) |

Таким образом из (7) видно, что на четвертом шаге итерации полученное значение совпадает с округленным значением на третьем шаге, а значит число 83 действительно является целочисленным кубическим корнем из числа 571787 и соответствует исходному сообщению *M=*83.

Выбор малой экспоненты шифрования *e* вообще говоря, допустим, но для защиты от соответствующей атаки необходимо «подсаливать» сообщения [ ]. Эффективным также является использование экспоненты *e*=216+1=65537, что требует 16 возведений в квадрат и одного умножения. Данный метод даже без «подсаливания» сообщений имеет преимущество перед выбором экспоненты *e*=3, поскольку используемая для малых экспонент атака будет эффективной, если только одно и то же сообщение шифруется и посылается одновременно 65537 пользователям, что маловероятно на практике.

## 2.2 Атака при малом объеме возможных сообщений

Атака при переборном числе сообщений довольно тривиальна и является возможной, если передаваемые сообщения не «подсаливаются».

Допустим, имеется набор сообщений  известных злоумышленнику при обозримом числе сообщений *r* (Это могут быть, например, различные команды – вперед, назад, влево, вправо и т.п.). Тогда перехваченная криптограмма может быть легко расшифрована.

Действительно, пусть злоумышленник перехватил криптограмму *C*. Для принятия решения о том, какое сообщение было передано, злоумышленник последовательно зашифровывает все варианты сообщений до появления совпадения с перехваченной криптограммой.



Способ борьбы с такой атакой – это «подсаливание» сообщений, т.е. присоединение к ним небольших псевдослучайных цепочек бит [ ].

## 2.3 Атака на малую секретную экспоненту

Атака на криптосистему РША с малой величиной секретной экспоненты, называемая также атакой Винера [ ], демонстрирует опасность, которая может возникнуть при проектировании системы с малой экспонентой дешифрования. Такой ситуацией, при которой использование малых экспонент является выгодным, может являться взаимодействие двух устройств с разницей в их вычислительной мощности. Примером является криптосистема РША, используемая во взаимодействии между смарт-картой и компьютером. В этом случае было бы предпочтительным использование малой секретной экспоненты для смарт-карты и малой открытой экспоненты для компьютера, чтобы уменьшить время вычислений, затрачиваемое смарт-картой.

В качестве математического базиса атака Винера использует элементы теории непрерывных дробей для нахождения в полиномиальной время некой дроби, достаточно близкая оценка которой известна злоумышленнику []. Атака Винера демонстрирует, как открытая экспонента *e* и модуль криптосистемы *n*=*pq* могут быть использованы для создания оценки некой дроби, которая содержит в себе секретную экспоненту *d*. Затем алгоритм, основанный на теории непрерывных дробей, использует известную оценку, составленную из открытого ключа, для нахождения секретной экспоненты *d*.

### 2.3.1 Необходимые сведения о непрерывных дробях

Непрерывной дробью является выражение вида:



В конкретной ситуации нас интересуют непрерывные дроби, которые имеют все , где . Для удобства будем обозначать:

|  |  |
| --- | --- |
|  | (8) |

Например,  означает, что  является разложением в непрерывную дробь для рационального числа 4/11.

*i*-ой подходящей дробью для непрерывной дроби  называется конечная непрерывная дробь , значение которой равно некоторому рациональному числу , где .

Разложение в непрерывную дробь для положительного рационального числа *f* выполняется путем вычитания целой части числа *f*, затем переворачивания оставшейся дробной части и вновь вычитания целой части до тех пор, пока дробная часть не будет равна 0 [ ]. Обозначим  - целое частное и  - дробная часть на шаге *i*, и обозначим через *m* последний шаг, на котором дробная часть оказалась равна нулю. Тогда:

|  |  |
| --- | --- |
| , ,  ,  для *i* = 1, 2, … . | (9) |

Поскольку рациональные числа всегда могут быть представлены в виде конечной непрерывной дроби, то на определенном шаге  обязательно будет равно нулю, а разложение в непрерывную дробь рационального числа *f* будет выглядеть, как.

На данном этапе могут быть сделаны два наблюдения, которые будут полезны в будущем. Первое –  (это действительно так, поскольку  означало бы, что , а это невозможно). Второе – для любого  справедливо:

|  |  |
| --- | --- |
| если m четное  если m нечетное | (10) |

В этом можно убедиться, глядя на число уровней дроби, вложенных в (1).

Теперь рассмотрим, как можно восстановить рациональное число *f* из разложения в непрерывную дробь. Используя формулу **(1).**, можно восстановить число *f*, начав с , суммируя и переворачивая полученную дробь на каждом шаге вплоть до . Однако, может оказаться более выгодным восстановление исходной дроби в прямом порядке, начиная с . Такой подход будет более удобным, к примеру, для последовательного восстановления рациональных чисел из соответствующих подходящих дробей, начиная с . Обозначим числитель и знаменатель рационального числа, как  и  соответственно. Тогда:

|  |  |
| --- | --- |
| и  для  [] | (11) |

Числитель и знаменатель рационального числа вычисляются по следующим формулам [ ]:

|  |  |
| --- | --- |
| , ,  , ,  ,  для *i* = 2, 3, … . | (12) |

В таком случае рациональное число  может быть восстановлено в прямом порядке, начиная с , как .

Существует взаимосвязь между числителем и знаменателем, которая окажется полезной в дальнейшем [ ]:

|  |  |
| --- | --- |
| для *i* = 2, 3, … . | (13) |

Алгоритм нахождения числителя и знаменателя некой дроби, для которой известно достаточно хорошее приближенное значение, будем называть «Алгоритмом непрерывных дробей».

**2.3.2 «Алгоритм непрерывных дробей»**

Обозначим через  «недооценку» рационального числа :

|  |  |
| --- | --- |
| для некоторого , | (14) |

Обозначим через ,  и ,  *i*-ые частные и дробные части чисел  и  соответственно. Если  достаточно мало, то числитель и знаменатель числа  может быть найден с использованием следующего алгоритма (повторяется до тех пор, пока  не найдено) [**Ошибка! Источник ссылки не найден.**]:

* Вычислить следующее целое частное () для разложения в непрерывную дробь числа 
* Использовать формулы **Ошибка! Источник ссылки не найден.** для восстановления *i*-ого рационального числа из подходящей дроби:  
   если *i* четное,  
   если *i* нечетное.
* Проверить, не является ли восстановленное на предыдущем шаге рациональное число интересующим нас числом.

Причиной добавления единицы к последнему  на четном шаге является тот факт, что строящееся предположение  должно быть больше, чем , так как по условию  является «недооценкой» числа , однако из **Ошибка! Источник ссылки не найден.** видно, что на четном шаге  меньше, чем .

Заметим, что для выполнения алгоритма должен существовать тест для проверки, является ли предположение о числе  верным.

Алгоритм непрерывных дробей приводит к решению [**Ошибка! Источник ссылки не найден.**], если:

|  |  |
| --- | --- |
| при четных  при нечетных m | (15) |

Теперь рассмотрим влияние условия **Ошибка! Источник ссылки не найден.** на максимально возможную величину . Решение уравнения **Ошибка! Источник ссылки не найден.** относительно  дает:

|  |  |
| --- | --- |
|  | (16) |

Ниже будет выполнен отдельный анализ для следующих случаев: , , четное  и нечетное .

Случай 1: 

Использование **Ошибка! Источник ссылки не найден.** для замены  в **Ошибка! Источник ссылки не найден.** дает:



С учетом **Ошибка! Источник ссылки не найден.** предыдущее неравенство упрощается до , что в свою очередь может быть переписано следующим образом (вспомним, что  и ):



Случай 2: 

Использование **Ошибка! Источник ссылки не найден.** для замены  в **Ошибка! Источник ссылки не найден.** дает:



С учетом **Ошибка! Источник ссылки не найден.** предыдущее неравенство упрощается до:

|  |  |
| --- | --- |
|  | (17) |

Ранее было замечено, что . Для данного случая это означает, что . Объединяя это с **Ошибка! Источник ссылки не найден.** и выражениями для  и  в **Ошибка! Источник ссылки не найден.** получим:



что является достаточным условием для гарантированной работы алгоритма непрерывных дробей.

Случай 3: четное 

Использование **Ошибка! Источник ссылки не найден.** для замены  в **Ошибка! Источник ссылки не найден.** дает:

|  |  |
| --- | --- |
|  | (18) |

С учетом **Ошибка! Источник ссылки не найден.** имеем:

 и



Подстановка этих выражений в **Ошибка! Источник ссылки не найден.** дает:



С учетом **Ошибка! Источник ссылки не найден.** и выражениями для  и  в **Ошибка! Источник ссылки не найден.** получаем неравенство:



Таким образом:



является достаточным для выполнения алгоритма непрерывных дробей.

Случай 4: нечетное 

Выполняя похожий анализ, что и в случае 3, получаем:



Поскольку  и , имеем . Таким образом,



является достаточным условием для выполнения алгоритма непрерывных дробей.

Принимая во внимание результаты анализа для всех четырех случаев, получаем неравенство:

|  |  |
| --- | --- |
|  | (19) |

что является достаточным условием для выполнения алгоритма непрерывных дробей. Напомним, что  и  являются числителем и знаменателем рационального числа , вычисляемые по формулам **Ошибка! Источник ссылки не найден.**.

### 2.3.3 Применение «алгоритма непрерывных дробей» для нахождения малой секретной экспоненты в РША.

Теперь рассмотрим, как полученные ранее сведения о непрерывных дробях и рассмотренный алгоритм непрерывных дробей может быть применен для криптоанализа РША. Из описания работы криптосистемы РША известно, что между открытым и секретным ключом существует следующее соотношение:



Из этого следует, что существует такое K, что:

|  |  |
| --- | --- |
|  | (20) |

Если обозначить , то по формуле **Ошибка! Источник ссылки не найден.**:

,

и тогда:



Возможно также, что K и G имеют общие множители. Тогда обозначим  и , а значит  и при этом . Теперь имеем:

|  |  |
| --- | --- |
|  | (21) |

Деление правой и левой части равенства **Ошибка! Источник ссылки не найден.** на  дает:

|  |  |
| --- | --- |
| , где | (22) |

Заметим, что соотношение  состоит полностью из известной информации об открытом ключе и является близкой оценкой соотношения  при выполнении условия **Ошибка! Источник ссылки не найден.** для . Перед тем, как применить алгоритм непрерывных дробей, нужно помнить, что он находит несократимую дробь, то есть числитель и знаменатель полученной дроби не будет иметь общих множителей, что показано в **Ошибка! Источник ссылки не найден.**. Из **Ошибка! Источник ссылки не найден.** мы видим, что . Поскольку k делит K, справедливо утверждение . Также известно, что  по определению. Таким образом выходит, что  и алгоритм непрерывных дробей может быть успешно использован для нахождения несократимой дроби  при условии, что  достаточно мало **Ошибка! Источник ссылки не найден.**.

Используя выражение для  из **Ошибка! Источник ссылки не найден.** и ограничение, наложенное на  в **Ошибка! Источник ссылки не найден.**, можно показать, что:

|  |  |
| --- | --- |
|  | (23) |

является достаточным для выполнения алгоритма непрерывных дробей. Заметим, что  в выражении **Ошибка! Источник ссылки не найден.** было упущено, поскольку является малым в сравнении с (*p*+*q*). Тем не менее это не влияет на верность утверждения **Ошибка! Источник ссылки не найден.**, так как  способствует лишь уменьшению .

Напомним, что для выполнения атаки необходимо реализовать алгоритм проверки. Рассмотрим, как можно проверить, действительно ли полученное на определенном шаге рациональное число является является дробью . В целях упрощения проверки будем считать, что . Это предположение не является значительно ограничивающим возможность проверки, потому что когда либо *e*, либо *d* заранее задано, ожидаемое значение обратной экспоненты приблизительно [**Ошибка! Источник ссылки не найден.**] (напомним, что ), и до тех пор, пока *G* выбрано большим, весьма вероятно, что . Из выражения **Ошибка! Источник ссылки не найден.** следствием того, что , является . Переписав выражение **Ошибка! Источник ссылки не найден.**, как:

|  |  |
| --- | --- |
|  | (24) |

нетрудно увидеть, что деление  на  дает целое частное

|  |  |
| --- | --- |
|  | (25) |

и остаток

|  |  |
| --- | --- |
| , | (26) |

так как . Таким образом мы можем найти предположения о  и о . Если вычисленное на определенном шаге предположительное  равно нулю, то предположительные  и неверны. Данный случай должен быть отброшен, поскольку это предполагает, что сомножителями модуля криптосистемы являются тривиально  и 1, чего быть не может. Предположение о  может быть использовано для вычисления предположительного , используя следующее равенство [**Ошибка! Источник ссылки не найден.**]:

|  |  |
| --- | --- |
|  | (27) |

Если предположение о  не является целым числом, то предположительные  и  не верны. Предположение о  может быть использовано далее для вычисления предположительного  используя следующее равенство [**Ошибка! Источник ссылки не найден.**]:

|  |  |
| --- | --- |
|  | (28) |

Если предположительное  является квадратом целого числа, то предположительные  и  верны. Тогда секретная экспонента  вычисляется делением  на  (напомним, что из **Ошибка! Источник ссылки не найден.** следует, что  - остаток от деления  на ).

Обозначим получаемые предположения из **Ошибка! Источник ссылки не найден.** и **Ошибка! Источник ссылки не найден.**, как:

|  |  |
| --- | --- |
|  | (29) |

Тогда при условии, что предположительное отношение  на текущем шаге является верным, можно получить факторизацию модуля криптосистемы, решая систему с двумя неизвестными **Ошибка! Источник ссылки не найден.**:



Если не предпринимать меры против данной атаки на РША, то ожидаемо, что  мало и . Из **Ошибка! Источник ссылки не найден.** видно, что при выполнении этих условий, секретная экспонента *d* битовой длины приблизительно до четверти битовой длины модуля может быть найдена в полиномиальное время [**Ошибка! Источник ссылки не найден.**].

Данная атака не может быть обобщена на обычный случай, когда секретная экспонента имеет приблизительно ту же битовую длину, что и модуль криптосистемы. Это происходит потому, что данная атака опирается на открытую экспоненту, содержащую в себе информацию, которая помогает факторизовать модуль [**Ошибка! Источник ссылки не найден.**]. В обычном же случае открытая экспонента может быть выбрана практически независимо от модуля.

### 2.3.4 Пример выполнения атаки на малую секретную экспоненту

Перед тем, как непосредственно привести пример, запишем последовательность необходимых действий:

Шаг 1. Записать открытый ключ в виде отношения  и обозначить *i*=0.

Шаг 2. Вычислить  и  для рационального числа , используя формулы **Ошибка! Источник ссылки не найден.**.

Шаг 3. Вычислить предположение о  по формулам **Ошибка! Источник ссылки не найден.** из разложения:

 если i четное,

 если i нечетное.

Шаг 4. Вычислить предположение о , умножив известную открытую экспоненту *e* на предположения о *dg* из шага 3.

Шаг 5. Вычислить предположение о , как целую часть от деления  на **Ошибка! Источник ссылки не найден.**, и предположение о g, как остаток от деления  на **Ошибка! Источник ссылки не найден.**. Если предположение , вернуться на шаг 2 и произвести вычисления для следующего *i*.

Шаг 6. Вычислить предположение о , воспользовавшись формулой **Ошибка! Источник ссылки не найден.** и произвести проверку: является ли  целым числом. Если нет, то предположение о  не является вреным, следует вернуться на шаг 2 и произвести вычисления для следующего *i*.

Шаг 7. Вычислить предположение о , воспользовавшись формулой **Ошибка! Источник ссылки не найден.** и произвести проверку: является ли  квадратом целого числа. Если нет, то предположение о  не является вреным, следует вернуться на шаг 2 и произвести вычисления для следующего *i*.

Шаг 8. Вычислить d путем деления знаменателя рационального числа  на . Также, если необходимо, получаем факторизацию модуля криптосистемы, решая систему уравнения с двумя неизвестными **Ошибка! Источник ссылки не найден.**.

Заметим также, что выполнение атаки прекращается на шаге 2 при неком *i*, если . Это означает, что данная атака не может быть применена при заданных условиях.

Приведем численный пример. Допустим, имеем секретную экспоненту *e*=2621 и модуль криптосистемы *n*=8927. Теперь покажем, как найти секретную экспоненту *d*, применяя вышеописанные шаги к рациональному числу . Результаты представим в виде таблицы (табл. 2.1).

Таблица 2.1

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Вычисляемая величина | Этап выполнения |  |  |  |
|  | Шаг 2 | 0 | 3 | 2 |
|  | Шаг 2 |  |  |  |
| предположение о | Шаг 3 |  |  |  |
| предположение о | Шаг 4 | 2621 | 7863 | 26210 |
| предположение о | Шаг 5 | 2621 | 7863 | 8736 |
| предположение о | Шаг 5 | 0 | 0 |  |
| предположение о | Шаг 6 | 3153.5 (нечетное, возврат к шагу 2) | 532.5 (нечетное, возврат к шагу 2) | 96 (четное) |
| предположение о | Шаг 7 |  |  | 289=172 (289 является квадратом целого числа) |
| секретная экспонента *d* | Шаг 8 |  |  | 5 |

Таким образом, при *i*=3 вычисленное предположение о  оказалось верным, а значит все последующие предположения тоже являются верными. Из знаменателя рационального числа  на шаге 8 была получена секретная экспонента *d*=5. Используя  и  дополнительно получаем простые сомножители модуля криптосистемы *p*=96+17=113 *q*=96-17=79.

### 2.3.5 Способы борьбы с атакой на малую секретную экспоненту

Существует 2 способа уменьшения максимального размера секретной экспоненты, которая может быть гарантированно найдена при помощи алгоритма непрерывных дробей. Из выражения **Ошибка! Источник ссылки не найден.** можно видеть, что противостоять атаке можно, увеличивая *k* и *g*.

Для увеличения *k* необходимо увеличивать открытую экспоненту *e* (см. равенство **Ошибка! Источник ссылки не найден.**). Этого можно достичь прибавлением  к экспоненте *e* [**Ошибка! Источник ссылки не найден.**].

Допустим, . Это влечет за собой  (см. равенство **Ошибка! Источник ссылки не найден.**). Замена  в **Ошибка! Источник ссылки не найден.** приводит к . Таким образом, если , успешное выполнение алгоритма непрерывных дробей при любой длине секретной экспоненты *d* не гарантировано. Увеличение длины открытой экспоненты *e* имеет свой недостаток - это увеличение времени выполнения операции шифрования на открытом ключе, хотя такая ситуация может быть приемлема при проектировании некоторых систем [**Ошибка! Источник ссылки не найден.**].

В свою очередь, как следует из определения , а значит для увеличения значения числа *g* сомножители *p* и *q* должны быть выбраны такими, чтобы  было большим.

## 2.4 Атака с использованием мультипликативного свойства шифра РША

Атака с использованием мультипликативного свойства шифра РША позволяет расшифровать криптограмму при условии, что легитимный пользователь криптосистемы согласен расшифровать любую другую криптограмму кроме той, что интересует злоумышленника.

Для любых сообщений *M1* и *M2* справедливо следующее [**Ошибка! Источник ссылки не найден.**]:

,

где , . Это свойство может использовать злоумышленник.

Предположим, что злоумышленник хочет дешифровать криптограмму *C*, предназначенную для легитимного пользователя, и предположим, что этот пользователь согласен дешифровать любую другую криптограмму для злоумышленника, кроме криптограммы *C*. Тогда для расшифровки криптограммы *C* злоумышленник выбирает случайное число *x* такое, что gcd(*x*, *n*) = 1. Затем вычисляет

|  |  |
| --- | --- |
|  | (30) |

и просит легитимного пользователя расшифровать данную криптограмму. В результате расшифровки получается бессмысленное на первый взгляд сообщение, но содержащее в себе интересующее злоумышленника исходное сообщение *M*:

|  |  |
| --- | --- |
|  | (31) |

Теперь, зная *x* и (31) для получения *M* достаточно вычислить:

|  |  |
| --- | --- |
|  | (32) |

Рассмотрим простой пример. Допустим, злоумышленник хочет расшифровать криптограмму *C* = 354. Открытый ключ, с помощью которого была вычислена данная криптограмма, (*e, n*) = (293, 361). Тогда злоумышленник выбирает любое целое число *x* (например, *x =* 283), которое является взаимно простым с модулем криптосистемы *n*, а значит существует обратный элемент от *x* по модулю *n*:

.

Далее злоумышленник вычисляет  по формуле (30):



и просит легитимного пользователя криптосистемы расшифровать . При условии, что легитимный пользователь согласился расшифровать , используя свой закрытый ключ (*d, n*) = (209, 361), злоумышленник получает некое сообщение  и вычисляет *M* по формуле (32):



Используя открытый ключ, нетрудно убедиться в том, что сообщению *M*=84 действительно соответствует криптограмма *C*=354.

Абсолютно аналогичная атака находит применение при использовании цифровой подписи на основе криптосистемы РША, когда злоумышленник просит легитимного пользователя подписать на первый взгляд бессмысленное сообщение, а в результате получает подпись на необходимое сообщение, воспользовавшись мультипликативным свойством алгоритма РША [**Ошибка! Источник ссылки не найден.**].

## 2.5 Циклическая атака

Предположим, что злоумышленнику известна криптограмма *C*. Циклическая атака [**Ошибка! Источник ссылки не найден.**] на РША представляет собой повторное выполнение операции шифрования над криптограммой:

; ; 

Повторное шифрование выполняется до тех пор, пока результат не совпадет с исходной криптограммой: . Тогда  и будет являться исходным сообщением. Данное событие рано или поздно произойдет на каком-то шаге *k*, поскольку шифрование – это по существу перестановка чисел . Обобщенная циклическая атака приводит также к факторизации модуля *n*, поэтому при больших *n* данный подход не лучше прямого метода факторизации модуля криптосистемы РША [**Ошибка! Источник ссылки не найден.**].

## 2.6 Атака на общие модули

Возможна ситуация, при которой несколько или все пользователи сети имеют в составе своего ключа одинаковый общий модуль *n*, но различные экспоненты *e* и *d*. Такое встречается при генерировании пар ключей пользователей некоторым общим «центром распределения ключей» [**Ошибка! Источник ссылки не найден.**]. Однако, существует вероятностный алгоритм, воспользовавшись которым любой пользователь *i*, имеющий в составе своих ключей экспоненты  и , способен факторизовать модуль  [**Ошибка! Источник ссылки не найден.**]. Тогда, если пользователь *j* имеет такой же модуль, то пользователь *i* с легкостью вычислит секретную экспоненту , используя соотношение **Ошибка! Источник ссылки не найден.**.

Покажем теперь, как знание обеих экспонент позволяет выполнить факторизацию модуля криптосистемы РША. Рассмотрим уравнение:

|  |  |
| --- | --- |
| , | (33) |

где *n* = *pq*. Так как число *n* является составным, уравнение (33) имеет четыре решения, два из которых могут быть использованы для нахождения сомножителей числа *n* [**Ошибка! Источник ссылки не найден.**]. Двумя тривиальными решениями называются , а двумя нетривиальными ≢. Покажем, почему это так. Рассмотрим равенство:

|  |  |
| --- | --- |
| , | (34) |

где *p* – простое число. Данное уравнение имеет всего 2 решения .

Аналогичным образом уравнение:

|  |  |
| --- | --- |
| , | (35) |

где *q* – простое число, имеет также два решения . С учетом того, что *n* = *pq*, по китайской теореме об остатках соотношение (33) будет верным только тогда, когда верны (34) и (35). Таким образом, уравнение (33) будет иметь 4 решения, которые могут быть найдены путем комбинирования решений равенств (34) и (35) четырьмя различными способами:

|  |  |
| --- | --- |
|  | (36) |
|  | (37) |
|  | (38) |
|  | (39) |

Комбинации решений (36) и (37) приводят к тривиальному решению уравнения (33) . Две последние комбинации (38) и (39) приводят к нетривиальному решению уравнения (33) ≢. Из (33) видно, что , но если был найден нетривиальный корень, то из (38) и (39) видно, что  или  не могут одновременно делиться на *p* и *q*. Тогда одно из чисел  или  делиться на *p*, а другое на *q* и нетривиальные сомножители числа n могут быть найдены, как :

|  |  |
| --- | --- |
|  | (40) |

Идея нахождения таких *x* и *y*, что , но ≢, используется многими известными методами факторизации, и в большинстве случаев разница заключается лишь в способах нахождения *x* и *y* [**Ошибка! Источник ссылки не найден.**]. Теперь рассмотрим свойства РША, которые позволяют найти такие *x* и *y*, зная обе экспоненты *e* и *d*.

Предположим, что мы имеем , где *n* – модуль криптосистемы. Тогда, если , то простые сомножители находятся аналогично (40) и в данном случае будут равны . Покажем, как можно найти такое число *x*. Из теоремы Эйлера известно, что для любого числа  взаимно простого с *n* справедливо:

|  |  |
| --- | --- |
|  | (41) |

С другой стороны, из **Ошибка! Источник ссылки не найден.** известно, что открытая и закрытая экспоненты в РША связаны следующим соотношением:



Отсюда следует, что  кратно φ. Значит можно заменить  на  в (41):

|  |  |
| --- | --- |
|  | (42) |

Также для удобства выполнения факторизации запишем  в виде:

|  |  |
| --- | --- |
|  | (43) |

где *k* и *t* являются положительными целыми числами, причем *t -* нечетное число.

Теперь, если для случайно выбранного числа *a* взаимно простого с модулем *n*, существует 0 < *k'* < *k* такое, что:

|  |  |
| --- | --- |
| и | (44) |

то *x* является нетривиальным квадратным корнем из единицы и делители числа *n* находятся, как . Данный алгоритм факторизации является вероятностным, потому что приблизительно для половины выбранных *a* будет найдено такое *k'*, что выполняется условие (44) [**Ошибка! Источник ссылки не найден.**].

Итак, запишем по порядку все шаги, необходимые для выполнения данного алгоритма:

Шаг 1. Привести к виду, показанному в (43).

Шаг 2. Выбрать случайное положительное целое число *a* > 1. Если случайное *a* окажется не взаимно простым с *n*, то делитель *n* вычисляется, как  и работа алгоритма завершается.

Шаг 3. Вычислить . Если результат вычисления равен 1 или , вернуться к шагу 2 и выбрать другое число *a*.

Шаг 4. Последовательно вычислять , где *i*=1, 2, 3 … k до тех пор, пока . Если на каком-то шаге , вернуться к шагу 2 и выбрать другое *a*.

Шаг 5. Как только  на определенном шаге *i*, вычислить делители модуля *n*, как .

В [**Ошибка! Источник ссылки не найден.**] доказывается, что данный алгоритм приводит к успеху как минимум в половине случаев при случайно выбранном числе *a*.

Для наглядности приведем небольшой пример. Допустим, имеется модуль криптосистемы *n*=2773, открытая экспонента *e*=17 и секретная экспонента *d*=157.

На первом шаге представим  в виде произведения нечетного числа и степени двойки, как показано в (43):

,

где нечетное *t*=667 и *k*=2.

Перейдем к шагу 2 и выберем в качестве *a* число *a*=7. На шаге 3 вычислим:



Поскольку на данном шаге результат вычисления  равен единице, необходимо вернуться к шагу 2 для выбора другого *a*. Возьмем тогда *a*=8. Вычислим:



Результат не равен 1 или *n*-1, тогда перейдем к шагу 4 и вычислим:

,

где *i*=1. Тот факт, что  и , означает, что 471 является нетривиальным квадратным корнем из единицы по модулю составного числа *n*. Тогда перейдем к шагу 5 и вычислим делители числа *n*:





Итак, использование общих модулей в криптосистеме РША приводит к возможности полного взлома криптосистемы любым из ее участников. Таким образом, использование общих модулей недопустимо.