Задание №4

Алгоритм RSA

Безопасность RSA основана на трудности разложения на множители больших чисел . Открытый и закрытый ключи являются функциями двух больших (100 - 200 разрядов или даже больше) простых чисел. Предполагается, что восстановление открытого текста по шифротексту и открытому ключу эквивалентно разложению на множители двух больших чисел .

Для генерации двух ключей используются два больших случайных простых числа , p и q. Для максимальной безопасности выбирайте p и q равной длины. Рассчитывается произведение:

$n = pq$

Затем случайным образом выбирается ключ шифрования e, такой что e и (p-1)(q-1) являются взаимно простыми числами. Наконец расширенный алгоритм Эвклида используется для вычисления ключа дешифрирования d, такого что

$ed=l(mod(p-l)(q-$

$l))$ Другими

словами

$d=e^{-1} mod((p-l)$

$(q-l))$

Заметим, что d и n также взаимно простые числа. Числа e и n - это открытый ключ, а число d - закрытый. Два простых числа p и q больше не нужны. Они должны быть отброшены, но не должны быть раскрыты .

Для шифрования сообщения m оно сначала разбивается на цифровые блоки, меньшие n (для двоичных данных выбирается самая большая степень числа 2, меньшая n). То есть, если p и q - 100-разрядные простые числа, то n будет содержать около 200 разрядов, и каждый блок сообщения то n, должен быть около 200 разрядов в длину. (Если нужно зашифровать фиксированное число блоков, их можно дополнить несколькими нулями слева, чтобы гарантировать, что блоки всегда будут меньше n. Зашифрованное сообщение c будет состоять из блоков c_t той же самой длины . Формула шифрования выглядит так

$c_{t=}\ m_i{}^e \bmod n.$

Для расшифровки сообщения возьмите каждый зашифрованный блок c_i, и вычислите

$m_i = c_t{}^{d} \bmod n$

Так как

$C^d = (m_i{}^e)^d = m_i{}^{ed} = m_i\text{-}^{k(p-1)(q-1)+1} = m_i\, m_i{}^{k(p-1)(q-1)} = m_i * 1 = m_i;$ все (mod n)

формула восстанавливает сообщение.

Табл. 1.

Шифрование RSA

Открытый ключ:

n произведение двух простых чисел p и q (p и q должны храниться в секрете)

e число, взаимно простое с $(p\text{-}1)(q\text{-}1)$

Закрытый ключ:

d $e^{-1} \bmod (p\text{-}1)(q\text{-}1)$

Шифрование:

$c = m^e \bmod n$

Дешифрирование:

$m = c^d \bmod n$

Точно также сообщение может быть зашифровано с помощью *d,* а расшифровано с помощью *e,* возможен любой выбор.
Короткий пример поможет пояснить работу алгоритма . Если $p = 47$ q = *11,* то

$n=pq=3337$

Ключ *e* не должен иметь общих множителей

$(p-1)(q-1)= 46*70 =3220$

Выберем (случайно) *e* равным 79. В этом случае $d= 79^{-1} \bmod 3220 = 1019$

При вычислении этого числа использован расширенный алгоритм Эвклида. Опубликуем *e* и *n,* сохранив в секрете *d.* Отбросим р и *q.* Для шифрования сообщения

m = 6882326879666683

сначала разделим его на маленькие блоки . Для нашего случая подойдут трехбуквенные блоки. Сообщение разбивается на шесть блоков m_i.

$m_1 = 688$

$m_2 = 232$

$m_3 = 687$

$m_4 = 966$

$m_5 = 668$

$m_6 = 003$

Первый блок шифруется как $688^{79} \bmod 3337 = 1570 = c_1$

Выполняя те же операции для последующих блоков, создает шифротекст сообщения :

c = 1570 2756 2091 2276 2423 158

Для дешифрирование нужно выполнить такое же возведение в степень, используя ключ дешифрирования 1019:

$1570^{1019} \bmod 3337 = 688 = m_1$

Аналогично восстанавливается оставшаяся часть сообщения .

Скорость RSA

Аппаратно RSA примерно в 1000 раз медленнее DES. Скорость работы самой быстрой СБИС-реализации RSA с 512-битовым модулем - 64 килобита в секунду . Существуют также микросхемы, которые выполняют 1024-битовое шифрование RSA. В настоящее время разрабатываются микросхемы, которые, используя 512-битовый модуль, приблизятся к рубежу 1 Мбит/с. Производители также применяют RSA в интеллектуальных карточках, но эти реализации медленнее.

Программно DES примерно в 100 раз быстрее RSA. Эти числа могут незначительно измениться при изменении технологии, но RSA никогда не достигнет скорости симметричных алгоритмов.

Табл. 2.
Скорости RSA для различных длин модулей при 8-битовом открытом ключе (на SPARC II)

	512 битов	768 битов	1024 бита
Шифрование	0.03с 0.16с	0.05с 0.48с	0.08с 0.93с 0.97с
Дешифрирование	0.16с 0.02с	0.52с 0.07с	0.08с
Подпись Проверка			

Программные Speedups

Шифрование RSA выполняется намного быстрей, если вы правильно выберете значение *e.* Тремя наиболее частыми вариантами являются 3, 17 и 65537 ($2^{16} + 1$). (Двоичное представление 65537 содержит только две единицы, поэтому для возведения в степень нужно выполнить только 17 умножений.) X.509

советует 65537, PEM рекомендует 3, а PKCS # - 3 или 65537. Не существует никаких проблем безопасности, связанных с использованием в качестве e любого из этих трех значений (при условии, что вы дополняете сообщения случайными числами), даже если одно и то же значение e используется целой группой пользователей.

Операции с закрытым ключом можно ускорить при помощи китайской теоремы от остатках, если вы сохр а-нили значения p и q, а также дополнительные значения: d mod $(p$ - 1), d mod $(q$ - 1) и ∂^{1} *modp* [1283, 1276]. Эти дополнительные числа можно легко вычислить по закрытому и открытому ключам .

Безопасность RSA

Безопасность RSA полностью зависит от проблемы разложения на множители больших чисел . Технически, это утверждение о безопасности некорректно. Предполагается, что безопасность RSA зависит от проблемы разложения на множители больших чисел. Никогда не было доказано математически, что нужно разложить n на множители, чтобы восстановить m по c и e. Понятно, что может быть открыт совсем иной способ криптоанализа RSA. Однако, если этот новый способ позволит криптоаналитику получить d, он также может быть использован для разложения на множители больших чисел.

Также можно вскрыть RSA, угадав значение $(p$-1$)(q$-1$)$. Это вскрытие не проще разложения n на множители .

Для сверхскептиков: доказано, что некоторые варианты RSA также сложны, как и разложение на множители . Раскрытие даже нескольких битов информации по зашифрованному RSA шифротексту не легче, чем дешифрирование всего сообщения .

Самым очевидным средством вскрытия является разложение n на множители. Любой противник сможет получить открытый ключ e и модуль n. Чтобы найти ключ дешифрирования d, противник должен разложить n на множители. В настоящее время передним краем этой технологии является число, содержащее 129 десятичных цифр . Значит, n должно быть больше этого значения.

Конечно, криптоаналитик может перебирать все возможные d, пока он не подберет правильное значение. Но такое вскрытие грубой силой даже менее эффективно, чем попытка разложить n на множители.

Вскрытие с выбранным шифротекстом против RSA

Некоторые вскрытия работают против реализаций RSA. Они вскрывают не сам базовый алгоритм, а надстроенный над ним протокол. Важно понимать, что само по себе использование RSA не обеспечивает безопасности. Дело в реализации.

Сценарий 1: Еве, подслушавшей линии связи Алисы, удалось перехватить сообщение c, шифрованное с помощью RSA открытым ключом Алисы. Ева хочет прочитать сообщение. На языке математики, ей нужно m, для которого

$m = c^{d}$

Для раскрытия m она сначала выбирает первое случайное число г, меньшее n. Она достает открытый ключ Алисы e. Затем она вычисляет

$x = f \bmod n$

$y = xc \bmod n$

$t = r^{-1} \bmod n$

Если $x = r^{e} \bmod n$, то $r = x^{d}$

mod n.

Теперь просит Алису подписать y ее закрытым ключом, таким образом расшифровав y. (Алиса должна подписать сообщение.) Не забывайте, Алиса никогда раньше не видела y. Алиса посылает Еве

$u=y^{d} \bmod n$

Теперь Ева вычисляет

$tu \bmod n = r^{-1} y^{d} \bmod n= r^{-}1x^{d}c^{d} \bmod n = c^{d} \bmod n = m$

И Ева получает *m.*

Никогда не пользуйтесь алгоритмом RSA для подписи случайных документов, подсунутых вам посторонними.

В ы в о д ы

— Знание одной пары показателей шифрования/дешифрирования для данного модуля позволяет взломщику разложить модуль на множители.

— Знание одной пары показателей шифрования/дешифрирования для данного модуля позволяет взломщику вычислить другие пары показателей, не раскладывая модуль на множители.

— В протоколах сетей связи, применяющих RSA, не должен использоваться общий модуль. (Это является быть очевидным следствием предыдущих двух пунктов.)

— Для предотвращения вскрытия малого показателя шифрования сообщения должны быть дополнены случайными значениями.

— Показатель дешифрирования должен быть большим.

Не забывайте, недостаточно использовать безопасный криптографический алгоритм, должны быть безопасными вся криптосистема и криптографический протокол. Слабое место любого из трех этих компонентов сделает небезопасной всю систему.

З а д а н и е .

1. Ознакомиться с алгоритмом RSA.
2. Разработать программу RSA.
3. Произвести шифрование контрольной фразы.
4. Оценить время шифрования и затраченные ресурсы компьтера.
5. Произвети расшифрование контрольной фразы.
6. Оценить затраченные ресурсы и время.