А.САНКИБАЕВ

ОБ АСИММЕТРИЧНОМ КРИПТОГРАФИЧЕСКОМ АЛГОРИТМЕ RSA

Нами поставлена цель разработать технологию информационной безопасности и криптографической защиты информации на основе односторонних функций для реализации эффективных асимметричных алгоритмов шифрования информации и формирования электронной цифровой подписи с заданной криптостойкостью [3].

Основные задания:

исследование возможностей применения однонаправленной функции с «секретом» для реализации асимметричного шифрования;

исследование функциональной правильности и криптостойкости криптографического алгоритма с открытым ключом;

исследование возможностей применения однонаправленной функции с «секретом» для формирования и проверки цифровой подписи.

Решить проблему распределения ключей без использования дорогостоящего, недоступного для противника канала связи долгое время считалось невозможным. Но в 1976 г. была опубликована работа американских ученых У. Диффи и

М. Э. Хеллмана «Новые направления в криптографии», в которой был предложен принципиально новый способ решения этой проблемы. Идеи У. Диффи и М. Э. Хеллмана не только существенно изменили криптографию, но и привели к появлению и бурному развитию новых направлений в математике. Центральным понятием «новой криптографии» является понятие односторонней функции [3].

Односторонней называется функция $F: X \to Y$, обладающая двумя свойствами:

а) существует полиномиальный алгоритм вычисления значений $F(x)$;

б) не существует полиномиального алгоритма инвертирования функции F (т. е. решения уравнения $F(x) = y$ относительно x).

Заметим, что односторонняя функция существенно отличается от привычных функций из-за ограничений на сложность ее вычисления и инвертирования. Вопрос о существовании односторонних функций пока открыт.

Еще одним новым понятием является понятие функции с секретом. Функцией с секретом K называется функция F_K: $X \to Y$, зависящая от параметра K и обладающая тремя свойствами:

1) существует полиномиальный алгоритм, который при любом K вычисляет значение $F_K(x)$;

2) не существует полиномиального алгоритма, который при неизвестном K инвертирует F_K;

3) существует полиномиальный алгоритм, который при известном K инвертирует F_K

Вопрос о существовании функций с секретом тоже пока открыт. Для практических целей криптографии было построено несколько функций, которые могут оказаться функциями с секретом. Для них свойство 2 пока строго не доказано, но считается, что задача инвертирования эквивалентна некоторой давно изучаемой трудной математической задаче. Наиболее известной и популярной из них является теоретико-числовая функция, на которой построен шифр RSA; в ней используется трудная задача разложения большого числа на множители.

Применение функций с секретом в криптографии позволяет:

а) Организовать обмен шифрованными сообщениями с использованием только открытых каналов связи, т. е. отказаться от секретных каналов связи для предварительного обмена ключами.

б) Включить в задачу вскрытия шифра трудную математическую задачу и тем самым повысить обоснованность стойкости шифра.

в) Решать новые криптографические задачи, отличные от шифрования (электронная цифровая подпись и др.).

Нашей задачей является создание односторонней функции с секретом с помощью нейронных сетей. Нейронные сети также представляют собой функцию с несколькими входами и одним выходом. Они успешно применяются для распознавания образов, создания систем искусственного интеллекта.

«Подпись вслепую», используемая в системе ecash, относится к так называемым «особым протоколам цифровой подписи», разрабатываемым гражданской и финансовой криптографией. Следует напомнить, что в современных криптографических системах, в том числе финансовых, используется так называемая технология «криптографии с открытым ключом». Надежность этой технологии основана на доказуемой эквивалентности задачи «взлома» криптосистемы какой-либо вычислительно сложной задаче. Например, при использовании одного из самых распространенных алгоритмов RSA каждый участник криптосистемы генерирует два случайных больших простых числа p и q, выбирает число e, меньшее pq и не имеющее общего делителя с (p–l)(q–1), и число d, такое, что (ed–l) делится на (p–l)(q–1). Затем он вычисляет n=pq, а p и q уничтожает [1]. Пара (n, e) называется «открытым ключом», а пара (n, d) – закрытым ключом». Открытый ключ передается всем остальным участникам криптосистемы, а закрытый сохраняется в тайне. Стойкость RSA есть функция сложности разложения произведения pq на простые множители p и q (эта задача решается для случая «вычисления закрытого ключа из открытого). При достаточной длине этих простых чисел (несколько тысяч двоичных разрядов) такое разложение вычислительно невозможно (т.е. требует ресурсов, недоступных в этом мире).

Для обеспечения конфиденциальности участник А «шифрует» сообщение m участнику Б с помощью открытого ключа Б: $c := m^e \bmod n$, а участник Б «расшифровывает его» с помощью своего закрытого ключа: $m := c^d \bmod n$. Для наложения «цифровой подписи» участник А «шифрует» сообщение m участнику Б с помощью своего закрытого ключа $s := m^d \bmod n$ и отправляет «подпись» s вместе с сообщением m. Участник Б может верифицировать подпись участника А с помощью открытого ключа А, проверив равенство[1].

Безопасность алгоритма RSA основана на трудоемкости разложения на множители (факторизации) больших чисел. Открытый и закрытый ключи являются функциями двух больших простых чисел разрядностью 100–200 десятичных цифр или даже больше. Предполагается, что восстановление открытого текста по шифртексту и открытому ключу равносильно разложению числа на два больших простых множителя [2].

Для генерации двух ключей применяются два больших случайных простых числа p и q, которые должны иметь равную длину. Рассчитывается произведение,

$$n = pq.$$

Затем случайным образом выбирается ключ шифрования e, такой, что e и (p–1)(q–1) являются взаимно простыми числами. Наконец, с помощью расширенного алгоритма Евклида вычисляется ключ расшифрования d, такой, что

$$ed = 1(\mathrm{mod}(p-1)(q-1)).$$

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

Рассмотрим пример, иллюстрирующий применение алгоритма RSA.

Зашифруем сообщение «CAB». Для простоты будем использовать маленькие числа (на практике применяются гораздо большие).

1. Выберем p = 3 и q = 11.

2. Определим n = 3·11 = 33.

3. Найдем (p–1)(q–1) = 20. Следовательно, в качестве d взаимно простое с 20, например, d=3.

4. Выберем число e. В качестве такого числа может быть взято любое число, для которого удовлетворяется соотношение (e·3) (mod 20) = 1, например 7.

5. Представим шифруемое сообщение как последовательность целых чисел с помощью отображения: A–1, B–2, C–3. Тогда сообщение принимает вид (3,1,2). Зашифруем сообщение с помощью ключа {7,33}:

ШТ1 = (3^7) (mod 33) = 2187 (mod 33) = 9,

ШТ2 = (1^7) (mod 33) = 1 (mod 33) = 1,

ШТ3 = (2^7) (mod 33) = 128 (mod 33) = 29.

6. Расшифруем полученное зашифрованное сообщение (9,1,29) на основе закрытого ключа {3,33}:

ИТ1 = (9^3) (mod 33) = 729 (mod 33) = 3,

ИТ2= (1^3) (mod 33) – 1 (mod 33) = 1,

ИТ3 = (29^3) (mod 33) – 24389 (mod 33) - 2.

Итак, в реальных системах алгоритм RSA реализуется следующим образом: каждый пользователь выбирает два больших простых числа и в соответствии с описанным выше алгоритмом выбирает два простых числа – e и d. Как результат умножения первых двух чисел (p и q) устанавливается n.

Далее {e, n} образует открытый ключ, а {d, n} – закрытый (хотя можно взять и наоборот).

Открытый ключ публикуется и доступен каждому, кто желает послать владельцу ключа сообщение, которое зашифровывается указанным алгоритмом. После шифрования сообщение невозможно раскрыть с помощью открытого ключа. Владелец же закрытого ключа без труда может расшифровать принятое сообщение.

В практических приложениях генерация простых чисел выполняется быстро [1].

1. Сгенерируйте случайное n-битовое число p.

2. Установите старший и младший биты равными 1 (старший бит гарантирует требуемую длину простого числа, а младший обеспечивает его нечетность).

3. Убедитесь, что p не делится на малые простые числа: 3, 5, 7, 11 и т.д. Во многих реализациях проверяется делимость p на все простые числа, меньшие 256. Наиболее надежна проверка делимости на все простые числа, меньшие 2000. Ее можно эффективно выполнить с помощью метода «колеса».

4. Выполняется тест Рабина–Миллера для некоторого случайного числа a. Если p проходит тест, генерируется другое случайное число a, и тест повторяется. Для ускорения вычислений выбираются небольшие значения a. Выполняются пять тестов. Если p не проходит один из тестов, то генерируется другое число p, и тест повторяется снова.

Этап 3 необязателен, но желателен. Проверка делимости случайного нечетного числа на 3, 5 и 7 отсекает 54 % нечетных чисел еще до этапа 4. Проверка делимости на все простые числа, меньшие 256, исключает 80 % нечетных чисел. В общем случае доля нечетных кандидатов, которые не делятся ни на одно простое число, меньшее n, равна 1.12/ln n. Чем больше тестируемое число n, тем больше предварительных вычислений нужно выполнить до теста Рабина–Миллера.

Перейдем непосредственно к изложению теста Рабина [3]. Проверим простоту входного чис-

ла m. Существует только одно четное простое число, равное 2. Поэтому любое простое число m, отличное от 2, является нечетным, а число m – 1 будет четным. Представим его в виде

$$m - 1 = 2^t \cdot s,$$

где $s < m$ – нечетное число, t=0,1,...

Выберем целое случайное число b, такое, что результат деления b на m не равен 0 или 1, т.е. $b \neq 0$, $b \neq 1 \pmod m$.

При выборе b используется датчик случайных чисел.

Использовав алгоритм быстрого возведения в степень по модулю m, вычислим следующую последовательность элементов кольца Z_m:

$x_0 = b^s \bmod m$,

$x_1 = x_0^2 \bmod m$,

$x_1 = x_1^2 \bmod m$,

...

$x^t = x_{t-1}^2 \bmod m = b^{m-1} \bmod m$.

(На каждом шаге возводим в квадрат число, полученное на предыдущем шаге.) Тест Рабина выдает ответ «m – составное число» в случае, если $x_t \neq 1 \pmod m$ (т. е. результат деления x_t на m равен единице) или в последовательности x_0, x_1, x_2,..., x_t имеется фрагмент вида ..., *, 1, ..., где звездочкой обозначено число, отличное от единицы или минус единицы по модулю m.

В противном случае тест Рабина выдает ответ «не знаю». Последовательность x_0, x_1 x_2,..., x_t в этом «плохом» случае либо начинается с единицы, либо содержит минус единицу где-нибудь не в конце.

Аппаратно реализованный алгоритм RSA работает примерно в 1000 раз медленнее алгоритма DES [3]. Существуют также микросхемы, которые выполняют 1024-битовое шифрование RSA. При программной реализации алгоритм DES работает примерно в 100 раз быстрее RSA. Эти числа могут незначительно измениться при изменении технологии, но RSA никогда не достигнет скорости работы симметричных алгоритмов.

Можно убедиться, что файл будет успешно зашифрован и расшифрован, но скорость этих операций очень мала: на компьютере с центральным процессором Intel Pentium III с тактовой частотой 1000 МГц шифрование протекает со скоростью 63,3 кБ/с, а дешифрование – со скоростью 4,5 кБ/с. Размер зашифрованного файла превышает размер незашифрованного файла на 9,4% (точнее, на 11/117).

Стойкость алгоритма RSA полностью зависит от трудоемкости решения проблемы разложения на множители больших чисел [1]. С технической точки зрения это не корректно. Утверждение, что безопасность RSA зависит от проблемы разложения на множители больших чисел, является гипотетическим. Математически не доказано, что для восстановления m по c и e необходимо разложить n на множители. Понятно, что может быть открыт совсем иной способ криптоанализа RSA. Но если этот новый способ позволит криптоаналитику получить d, он тоже может быть использован для разложения на множители больших чисел. Это не вызывает никаких опасений. Доказано также, что раскрытие даже нескольких битов информации из зашифрованного с помощью RSA шифртекста не легче, чем дешифрование всего сообщения.

Кроме того, можно вскрыть RSA, угадав значение (p–l)(q–l). Это вскрытие не проще разложения n на множители.

Самым очевидным средством вскрытия является разложение n на множители. Любой противник сможет получить открытый ключ e и модуль n. Чтобы найти ключ расшифрования d, противник должен разложить n на множители. В настоящее время последним достижением технических средств разложения на множители является число, содержащее 200 десятичных цифр. Значит, выбираемое число n должно быть больше этого значения.

В 1994 г. разложено на множители число, содержащее 129 десятичных знаков, с помощью алгоритма квадратичного решета. В 1999 г. разложено 140-значное число. Затем все ускорилось. RSA Laboratories опубликовала цепочку ключей возрастающей длины и назначила награду за «взлом» каждого следующего числа. Сейчас таких чисел восемь, причем последнее длиной в 2048 бит (оно записывается 617 десятичными знаками) оценили в 200 тыс. долларов. 200-значное разложено на множители в мае 2005 г. Новое достижение принадлежит той же группе исследователей из Бонна. Вычислителям потребовалось 80 2,2-гигагерцевых процессоров Opteron, которые были непрерывно загружены пять месяцев подряд [4].

На сайте rsasecurity.com очередную (и, надо заметить, предсказуемую) победу над ключом оценивают как поражение. Аргументы понятны: чтобы взломать не самый сложный шифр, все еще требуются месяцы работы современного вычислительного кластера. Любопытно другое. 80 процессоров – далеко не последнее слово техники: существуют гораздо более мощные суперкомпьютеры. Однако даже общедоступные BlueGene с 1024 процессорами, которые компания IBM продает по 2 млн долларов, ни разу за последнее время не упоминались в новостях, посвященных борьбе с алгоритмом RSA.

Имеются методы взлома, предназначенные для вскрытия реализаций алгоритмов RSA. Они вскрывают не сам базовый алгоритм, а использующий его протокол. Важно понимать, что само по себе использование RSA не обеспечивает безопасности. Дело еще и в деталях реализации [1].

Сценарий 1. Еве, подслушивавшей линии связи Алисы, удалось перехватить сообщение c, шифрованное с помощью RSA открытым ключом Алисы. Ева хочет прочитать сообщение. В математической форме ей нужно узнать *m*, для которого

$$m = c^d.$$

Для вскрытия m она сначала выбирает первое случайное число r, меньшее n, и достает открытый ключ Алисы e. Затем она вычисляет

$x=r^e \bmod n$,

$y=xc \bmod n$,

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

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

Теперь Ева просит Алису подписать у ее закрытым ключом, тем самым расшифровав y. (Алиса должна подписать сообщение, а не его хэш-значение). Необходимо помнить, что Алиса никогда раньше не видела y. Алиса посылает Еве

$$u=y^d \bmod n.$$

После этого Ева вычисляет

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

И Ева получает m.

Сценарий 2. Трент – это компьютер-нотариус. Если Алиса хочет заверить документ, она посылает его Тренту. Трент подписывает его цифровой подписью RSA и отправляет обратно. (Однонаправленные хэш-функции не используются, Трент шифрует своим закрытым ключом все сообщение.)

Мэллори хочет, чтобы Трент подписал такое сообщение, которое в обычном случае он никогда не подпишет. Это сообщение может содержать фальшивую временную метку, или же автором этого сообщения может являться другое лицо. Какой бы ни была причина, Трент никогда не подпишет это сообщение, если у него будет возможность выбора. Назовем это сообщение m'.

Сначала Мэллори выбирает произвольное значение x и вычисляет $y=x^e \bmod n$. Значение e он может получить без труда – это открытый ключ Трента, который должен быть опубликован, чтобы можно было проверять подписи Трента. Теперь Мэллори вычисляет значение $(m^d \bmod n)\ x^{-1} \bmod n$, которое равно $m^{rd} \bmod n$ и является m'.

На самом деле Мэллори может использовать множество методов решения подобной задачи. Слабым местом алгоритма, которое используют атаки такого рода, является сохранение мультипликативной структуры входа при возведении в степень или

$$(xm)^d \bmod n=x^d m^d \bmod n.$$

Таким образом, необходимо помнить следующее. Нельзя использовать алгоритм RSA для подписи случайных документов. Вначале всегда следует воспользоваться однонаправленной хэш-функцией.

ЛИТЕРАТУРА

1. *Шнайер Брюс.* Прикладная криптография. М.: Триумф, 2002.

2. *Чмора А. Л.* Современная прикладная криптография. М.: Гелиос АРВ, 2002. С. 72–82.

3. *Кононов И. А.* // Вестник СахГУ. 2005.

4. BYTE/Россия. № 5. 2004 г.

Резюме

RSA асимметриялық хаттамасы толық келтірілген. Біржақты функция мен цифрлық қолтаңба жөнінде толық та, түсінікті анықтама берілген. Жәй сандарды таңдау мен тексеру әдісі жазылған, Рабин-Миллер әдісі көрсетілген. RSA сенімділігі жан-жақты тексеріліп, оны шифр таңдау арқылы бұзу негізгі әдістері қарастырылған. RSA-мен шифрлеу мен шифрлерді шешу жылдамдық характеристикалары келтірілген. Бүгінгі күннің жаңа серверлер арқылы алгоритмді бұзатын жетістіктер қарастырылған.

Summary

In the article it is given detailed description of asymmetric cryptographic protocol RSA. There is entered clear determination to unilateral function and digital signature. The procedure to generation simple numbers and their checking for simplicity is described, it is described method of Rabin-Miller. Hereinafter detailed is considered stability RSA and methods of its breaking in on base selected ciphertext. Also we brought speed features of the encryption and decryption RSA. The last achievements is breaking in by method of the factorization by means of powerful modern multiprocessor server.

УДК 004.056

ЕНУ им. Л. Гумилева *Поступила 2.07.2006 г.*