22 RSA

22.1 Криптосистема RSA

Набор алгоритмов (Gen, Enc, Dec) — это, говоря как программисты, *интерфейс*, который требует *инстанциирования*. Один из способов такого инстанциирования был предложен Ривестом, Шамиром и Адлеманом, университетскими исследователями, в 1974 году. Впоследствии предложенная ими криптосистема стала называться RSA, по первым буквам фамилий авторов.

Алгоритмы RSA определяются следующим образом.

Алгоритм Gen (RSA)

Вход: 1^ℓ.

Выход: открытая экспонента e (долговременные параметры par), модуль n (открытый ключ pk), секретная экспонента d (личный ключ sk).

Шаги:

1. Выбрать натуральное нечетное $e \geqslant 3$.
2. $p, q \xleftarrow{R} PRIMES$: $p \neq q$, p, q — нечетные, $(e, p-1) = 1$, $(e, q-1) = 1$, $\lceil \log_2 p + \log_2 q \rceil = \ell$.
3. $n \leftarrow pq$ (n является ℓ-битовым числом).
4. Вычислить $\varphi(n) = (p-1)(q-1)$ [условия на p и q гарантируют, что $(e, \varphi(n)) = 1$].
5. Определить $d = e^{-1} \bmod \varphi(n)$.
6. Возвратить (e, n, d).

Позже мы обоснуем, что все использованные операции можно выполнить за полиномиальное время.

Возвращаемые алгоритмом данные именные: e — *открытая экспонента*, n — *модуль*, d — *секретная экспонента*. Обычно e фиксируется и, таким образом, является долговременным параметром. Модуль n — открытый ключ, секретная экспонента d — личный.

На практике $\ell = 512$ (уже ненадежно), $\ell = 1024$ (пока надежно), $\ell = 2048$ (надежно), $\ell = 4096$ (очень надежно). Стандарт NIST SP 800-57 указывает на соотношение между длиной модуля ℓ и паритетной длиной ключа симметричной криптосистемы (например, AES):

паритетная длина ключа	ℓ	паритетная длина ключа	ℓ
80	1024	192	7680
112	2048	256	15360
128	3072		

Модуль n определяет строение множеств открытых и шифртекстов: $\mathcal{X} = \mathcal{Y} = \mathbb{Z}_n$.

Алгоритм Enc (RSA)

Вход: n, e, $x \in \mathbb{Z}_n$ — открытый текст.

Выход: $y \in \mathbb{Z}_n$ – шифтекст.

Шаги:

1. $y \leftarrow x^e \bmod n$.
2. Возвратить y.

Алгоритм Dec (RSA)

Вход: n, d, y.

Выход: x.

Шаги:

1. $x \leftarrow y^d \bmod n$.
2. Возвратить x.

Отметим, что оба алгоритма являются детерминированными. Поэтому криптосистему RSA можно преобразовать в систему ЭЦП. Подпись документа $m \in \mathbb{Z}_n$ определяется как $s = m^d \bmod n$, а алгоритм Vfy состоит в проверке равенства:

$$s^e \bmod n \stackrel{?}{=} m.$$

Если текст $x \notin \mathbb{Z}_n$, т. е. $(x, n) \neq 1$, то можно без труда факторизовать n и, как мы увидим далее, атаковать RSA. Тем не менее, мы разрешаем текстам быть элементами $\mathbb{Z}_n \setminus \mathbb{Z}_n^*$. На практике к открытым текстам добавляются случайные данные.

22.2 Корректность RSA

Теорема 22.1 (корректность RSA). Криптосистема RSA корректна:

$$\mathtt{Dec}(n, d, \mathtt{Enc}(n, e, x)) = x$$

для любых (n, e, d), полученных с помощью алгоритма `Gen`, и всех $x \in \mathbb{Z}_n$.

Доказательство. По построению

$$ed \equiv 1 \pmod{\varphi(n)} \Rightarrow ed = r\varphi(n) + 1.$$

Если $x \in \mathbb{Z}_n^*$, то отсюда по теореме Эйлера немедленно следует нужный результат:

$$x^{ed} = \left(x^{\varphi(n)}\right)^r x \equiv x \pmod{n}.$$

Для общего $x \in \mathbb{Z}_n$ рассуждения становятся более тонкими. Если $x \not\equiv 0 \pmod{p}$, то по малой теореме Ферма

$$x^{p-1} \equiv 1 \pmod{p}.$$

Поэтому

$$x^{ed} = x^{r\varphi(n)+1} = x(x^{p-1})^{r(q-1)} \equiv x \pmod{p}.$$

Данное равенство очевидно выполняется также и для $x \equiv 0 \pmod{p}$, т. е. для всех x.

Аналогично, для всех x выполняется

$$x^{ed} \equiv x \pmod{q}.$$

Последние два равенства означают, что

$$x^{ed} \equiv x \pmod{n},$$

что и требовалось доказать. □

Пример 22.1 (несекретное засекречивание). В конце 1960-х годов Эллис из секретной службы Великобритании предложил схему «несекретного засекречивания», которая является прототипом сегодняшних криптосистем с открытым ключом. В 1973 году Кокс предложил реализацию данной схемы на базе задач теории чисел. Предложенная Коксом реализация «несекретного засекречивания» отличается от RSA только тем, что открытая экспонента e совпадает с модулем n. Очевидно, что при этом $(e, \varphi(n)) = 1$ автоматически. □

Пример 22.2 (мини RSA). Пусть $p = 11$, $q = 3$. Следовательно, $n = 33$ и $\varphi(n) = 20$. Для $e = 3$ выполняется

$$(e, p-1) = (10, 3) = (3, 1) = 1, \quad (e, q-1) = (3, 2) = (2, 1) = 1 \quad \Rightarrow \quad (e, \varphi(n)) = 1,$$

т. е. $e = 3$ является допустимой открытой экспонентой.

Секретная экспонента: $d = e^{-1} \bmod \varphi(n) = 7$.

Зашифрование $x = 5$: $y = 5^7 \bmod 33 = 26$. Расшифрование: $26^3 \bmod 33 = 5$. □

22.3 RSA и факторизация

Противнику Виктору требуется определить открытый текст по шифртексту при известных долговременных параметрах и открытом ключе. Конкретнее, Виктор сталкивается с решением следующей задачи.

Задача `RSA`	
Вход	n — модуль RSA, e — открытая экспонента RSA, $y \in \mathbb{Z}_n$.
Выход	$x \in \mathbb{Z}_n$: $x^e = y \bmod n$.

Решение x называется корнем e-й степени из y по модулю n:

$$x = y^{1/e} \bmod n.$$

Задача RSA — это задача извлечения корней e-й степени.

Задача RSA связана с задачей Factor следующим соотношением сводимости: RSA $\leqslant_P$ Factor. Действительно, если Виктор решил задачу факторизации $n \mapsto \{p, q\}$, то он может определить $\varphi(n) = (p-1)(q-1)$, затем вычислить $d = e^{-1} \mod \varphi(n)$ и найти $x = y^d \bmod n$.

Задача Factor хорошо изучена и признается трудной. Верно ли, что Factor $\leqslant_P$ RSA или Factor $\leqslant_R$ RSA? Если ответ на вопрос положительный, то задачи RSA и Factor (вероятностно) полиномиально эквиваленты, и RSA так же трудна как Factor, что является весомым аргументом в пользу стойкости RSA.

В связи с вопросом о сводимости Factor к RSA рассмотрим две промежуточные задачи.

Задача EulerPhi	
Вход	n — модуль RSA.
Выход	$\varphi(n)$.

Задача SecExp	
Вход	(n, e), где n — модуль RSA, $(e, \varphi(n)) = 1$.
Выход	$d = e^{-1} \mod \varphi(n)$.

Понятно, что

$$\texttt{RSA} \leqslant_P \texttt{EulerPhi} \leqslant_P \texttt{Factor}, \quad \texttt{RSA} \leqslant_P \texttt{SecExp} \leqslant_P \texttt{Factor}.$$

Оказывается также, что

1. Factor $\leqslant_P$ EulerPhi. Действительно, пусть известно значение $\varphi(n) = (p-1)(q-1)$ и пусть, не нарушая общности, $p > q$. Для определения p и q достаточно решить систему линейных уравнений:

   $$\begin{aligned} p + q &= n - \varphi(n) + 1, \\ p - q &= \sqrt{p^2 + q^2 - 2pq} = \sqrt{(p+q)^2 - 4n}. \end{aligned}$$

   Отметим, что вычисление квадратного корня в $\mathbb{Z}$ может быть проведено за полиномиальное время (в отличие от $\mathbb{Z}_n$).

   Задачу EulerPhi можно сформулировать для произвольных n. Для общей задачи удается доказать сведение Factor $\leqslant_R$ EulerPhi.

2. Factor $\leqslant_R$ SecExp: существует верятностный алгоритм типа Лас — Вегас, который находит $\{p, q\}$ по (n, e, d).

Сказанное означает, что задачи Factor, EulerPhi, SecExp полиномиально эквивалентны, и RSA сводится к каждой из них. Но нельзя сказать, что какая-либо из задач сводится к RSA. Нужное сведéние снова не установлено.

Трудности с доказательством сведéния Factor к RSA привели исследователей к мысли о том, что этого сведéния просто не существует. Наибольший прогресс в доказательстве последнего факта был достигнут Боне и Венкатесаном в 1998 г.

Теорема 22.2 (Боне — Венкатесан). Пусть e — мало и $M^{\texttt{RSA}}$ — алгоритм, который выполняет факторизацию n, используя полиномиальное число обращений к оракулу RSA: $y \mapsto y^{1/e} \bmod n$ и полиномиальное число арифметических операций. Тогда $M^{\texttt{RSA}}$ можно преобразовать в обычный (без обращений к RSA) полиномиальный алгоритм M', который выполняет факторизацию n.

Поскольку задача Factor считается трудной, алгоритм M' скорее всего не существует. Поэтому $M^{\texttt{RSA}}$ и сведение Factor $\leqslant$ RSA также скорее всего не существуют, по крайней мере, при малых e.

Сказанное отнюдь не означает, что задача RSA не является трудной. Она вполне может быть трудной, просто это *другая* трудная задача, не связанная с Factor.