Криптография с открытым ключом

Сергей Николенко

Криптография — CS Club, осень 2009

Outline

Арифметика по модулю n

- $\mathbb{Z}_n^+$ — это группа по сложению.
- $\mathbb{Z}_n^*$ — это группа по умножению.
- Сколько элементов в $\mathbb{Z}_n^*$?

Арифметика по модулю n

- $\mathbb{Z}_n^+$ — это группа по сложению.
- $\mathbb{Z}_n^*$ — это группа по умножению.
- Сколько элементов в $\mathbb{Z}_n^*$?
- Обратимые элементы в $\mathbb{Z}_n$ — это взаимно простые с n.
- Их всего $\phi(n)$ — функция Эйлера. Если p и q простые, то

$$\phi(p) = p - 1, \qquad \phi(pq) = (p-1)(q-1).$$

Арифметика по модулю n

- $\mathbb{Z}_n^+$ — это группа по сложению.
- $\mathbb{Z}_n^*$ — это группа по умножению.
- Сколько элементов в $\mathbb{Z}_n^*$?
- Если p — простое, то $\mathbb{Z}_p$ — это поле: у каждого элемента, кроме нуля, есть обратный по умножению.
- Над полем верны полезные факты из алгебры: например, над полем многочлен степени d имеет не более d корней.

Арифметика по модулю n

- На всякий случай ещё вспомним, что бывают конечные поля с p^m элементами.
- Их можно рассматривать как поля многочленов по модулю того или иного неприводимого многочлена.
- Например, поле $\mathbb{F}_{16}$ состоит из следующих элементов:

$$\begin{array}{llll} 0, & x^2 & x^3 & x^2+x^3 \\ 1 & x^2+1 & x^3+1 & x^2+x^3+1 \\ x & x^2+x & x^3+x & x^3+x^2+x \\ x+1 & x^2+x+1 & x^3+x+1 & x^3+x^2+x+1 \end{array}$$

- Операции производятся по модулю x^4+x+1 (или x^4+x^3+1, или $x^4+x^3+x^2+1$ — получится одно и то же поле).

Малая теорема Ферма

- Если p простое, то для любого a $a^p \equiv a \pmod{p}$, а для любого a, взаимно простого с p, $a^{p-1} \equiv 1 \pmod{p}$.
- Соответственно, для простого p и любых m и n

$$\text{если } m \equiv n \pmod{p-1}, \quad \text{то } \forall a \; a^m \equiv a^n \pmod{p}.$$

- Теорема Эйлера — для любого n и любого a, взаимно простого с n,

$$a^{\Phi(n)} \equiv 1 \pmod{n}.$$

Алгоритм Евклида

- Алгоритм Евклида: классический — вычисляет gcd.
- Кроме $d = \gcd(a, b)$, вычисляет ещё два числа x и y, такие, что $ax + by = d$.
- Как применить алгоритм Евклида, чтобы найти $a^{-1} \pmod{n}$?

Алгоритм Евклида

- Алгоритм Евклида: классический — вычисляет gcd.
- Кроме $d = \gcd(a, b)$, вычисляет ещё два числа x и y, такие, что $ax + by = d$.
- Как применить алгоритм Евклида, чтобы найти $a^{-1} \pmod{n}$?
- Найти такие x и y, что $ax + ny = d$, где $d = \gcd(a, n)$.
- Если $d > 1$, то a необратимо в $\mathbb{Z}_p$; если $d = 1$, то $x = a^{-1} \pmod{n}$.

Возведение в степень

- Если есть два числа a и b по модулю n, и мы хотим вычислить $a^b \pmod n$, то можно вычислить

$$a^2 \pmod n,\ a^3 \pmod n,\ \ldots$$

- Здесь $b - 1$ умножение по модулю n.
- Можно ли лучше?

Repeated squarings

- Можно сделать так: запишем b как строку битов. Потом будем возводить a в квадрат, домножая на a там, где у b биты равны 1. Например:

$$b = 9_{10} = 1001_2 \quad \Rightarrow \quad a^b = ((a^2)^2)^2 \cdot a, \text{ 4 умножения.}$$

$$b = 65537_{10} = 1000000000000001_2 \quad \Rightarrow$$
$$\Rightarrow \quad a^b = (((a^2)^2)\ldots)^2 \cdot a, \text{ 17 умножений.}$$

- 17 значительно меньше, чем 65536.

Квадратные корни

- Теперь давайте наоборот. Как по $x^2 \pmod p$ найти $x \pmod p$?
- Во-первых, не всякое число является квадратом по модулю p. Те, которые являются, называются *квадратичными вычетами*.
- В $\mathbb{Z}_p^*$ вычетов столько же, сколько невычетов, а именно $\frac{p-1}{2}$. Почему?

Квадратные корни

- Рассмотрим $1^2, 2^2, \ldots, \left(\frac{p-1}{2}\right)^2$.
- Поскольку $n^2 \equiv (p-n)^2 \pmod{n}$, всего вычетов не больше $\frac{p-1}{2}$.
- Пусть их меньше. Тогда для некоторых $1 \leq i, j \leq \frac{p-1}{2}$

$$i^2 \equiv (-i)^2 \equiv j^2 \equiv (-j)^2 \pmod{p}.$$

- Иначе говоря, у уравнения $x^2 \equiv i^2 \pmod{p}$ четыре разных корня.
- Но $\mathbb{Z}_p$ — поле, и у него не может быть больше двух корней.

Символ Лежандра

- Символ Лежандра:

$$\left(\frac{a}{p}\right) = \begin{cases} 0, & a \equiv 0 \pmod{p}, \\ 1, & a \not\equiv 0 \pmod{p}, \text{ и для некоторого } x\ x^2 \equiv 0, \\ -1, & a \not\equiv 0 \pmod{p}, \text{ и такого } x \text{ не существует.} \end{cases}$$

- Для простого p

$$\left(\frac{a}{p}\right) \equiv a^{(p-1)/2} \pmod{p}.$$

Символ Лежандра

- Кроме того,

$$\left(\frac{ab}{p}\right) = \left(\frac{a}{p}\right)\left(\frac{b}{p}\right),$$

$$\left(\frac{p}{q}\right) = (-1)^{\frac{p-1}{2}\frac{q-1}{2}}\left(\frac{q}{p}\right).$$

- Это позволяет построить алгоритм для вычисления символа Лежандра $\left(\frac{a}{p}\right)$:
  - разложить $\left(\frac{a}{p}\right)$ в произведение $\left(\frac{p_1}{p}\right)\ldots\left(\frac{p_m}{p}\right)$;
  - заменить на $p_i \pmod p$, перевернуть, повторить.

Квадратный корень

- Теперь возвращаемся к квадратному корню. Пусть дано простое p и $a \in \mathbb{Z}_p$.
- Если $p \equiv 3 \pmod 4$, то корень ищется как

$$x \equiv a^{(p+1)/4} \pmod n.$$

- Действительно,

$$1 = \left(\frac{a}{p}\right) \equiv a^{(p-1)/2} \pmod p.$$

- Значит,

$$x^2 \equiv a^{(p+1)/2} \equiv a \cdot a^{(p-1)/2} \equiv a \pmod p.$$

Квадратный корень

- Для $p \equiv 1 \pmod 4$ — вероятностный алгоритм.
- Рассмотрим многочлен $x^{(p-1)/2} - 1$. Он степени $\frac{p-1}{2}$, его корни — все квадратичные вычеты по модулю p, и только они.
- Теперь рассмотрим многочлен $f(x) \equiv x^2 - a \equiv (x - r)(x + r) \pmod p$. Подставим

$$f(x - \delta) \equiv (x - (\delta - r))(x - (\delta + r)) \pmod p.$$

- Факт (без доказательства): для половины δ одно из значений $(\delta - r)$, $(\delta + r)$ является вычетом, а другое — нет.
- Выберем δ случайно и подсчитаем $\gcd(f(x - \delta), x^{(p-1)/2} - 1)$ (как многочленов).
- Тогда с вероятностью $1/2$ мы получим корень из a.

Для составных n

- Пусть, например, $n = pq$. Алгоритм вычисления квадратного корня из a по модулю n.
  1. Найти корни $(r, -r)$ числа a по модулю p.
  2. Найти корни $(s, -s)$ числа a по модулю q.
  3. Найти алгоритмом Евклида такие c и d, что $cp + dq = 1$.
  4. Вычислить $x = rdq + scp \pmod{n}$ и $y = rdq - scp \pmod{n}$.
  5. Вернуть $(\pm x, \pm y)$.
- Иначе говоря, мы можем вычислять квадратные корни, если умеем раскладывать n на множители.

Обсуждение алгоритма

- Вычисление квадратного корня потребовало уметь раскладывать a на множители.
- Без этого даже не проверить, является ли a вычетом.
- А можно ли наоборот? Можно ли разложить число на множители, умея вычислять квадратные корни по его модулю?

Разложение на множители через $\sqrt{a}$

- Можно! Предположим, что мы умеем выдавать некий квадратный корень по модулю n.
- Возьмём случайное x, вычислим $a = x^2$ и подадим алгоритму.
- Если мы получили $\pm x$, повторим операцию. А если получили $y \neq \pm x$, то получилось, что

$$x^2 \equiv y^2 \pmod{n}, \text{ но } y \neq \pm x \pmod{n}.$$

- Это значит, что n делит $x^2 - y^2 = (x - y)(x + y)$, но при этом не делит либо $x - y$, либо $x + y$.
- Значит, $\gcd(x - y, n)$ — нетривиальный делитель n.

Постановка задачи

- Теперь поставим более сложную задачу — найти логарифм.
- *Дискретный логарифм*: по простому числу p, числу $a \in \mathbb{Z}_p^*$, порождающему $\mathbb{Z}_p^*$, и числу $b \in \mathbb{Z}_p^*$ найти такое $0 \leq x \leq p-2$, что

$$a^x \equiv b \pmod{p}.$$

- *Обобщённый дискретный логарифм*: то же в произвольной циклической группе G: по генератору $a \in G$ и $b \in G$ найти такой x, что $a^x = b$.

Замечания

- Сложность не зависит от генератора a; для другого генератора a'

$$a^x = b = a'^y = (a^z)^y, \text{ и } \log_{a'} b = \log_a b (\log_a a')^{-1}.$$

- Но сложность зависит от представления группы, т.е. для изоморфных групп сложность дискретного логарифма может быть разной. Почему?

Замечания

- Сложность не зависит от генератора a; для другого генератора a'

$$a^x = b = a'^y = (a^z)^y, \text{ и } \log_{a'} b = \log_a b (\log_a a')^{-1}.$$

- Но сложность зависит от представления группы, т.е. для изоморфных групп сложность дискретного логарифма может быть разной. Почему?
- Потому что любая циклическая группа изоморфна $\mathbb{Z}_n^+$ для некоторого n.
- Дискретный логарифм в $\mathbb{Z}_n^+$ — это значит найти такой x, что $ax = b \pmod{n}$. Наверное, это не так уж сложно...

Замечания

- Алгоритмы для задачи дискретного логарифма делятся на три группы:
  1. Работающие для любых групп.
  2. Работающие для любых групп, но эффективные для «гладких» (когда порядок группы имеет маленькие простые делители).
  3. Эффективные только для некоторых групп.
- Мы будем их изучать в этом курсе, но позже.

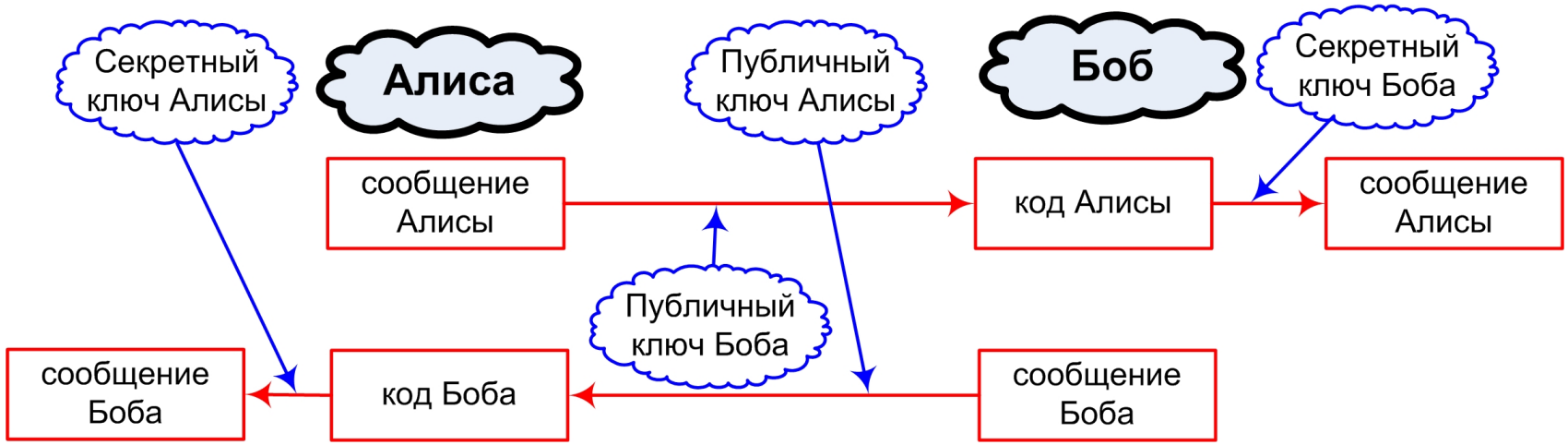
Итоги присказки

- Мы теперь умеем в $\mathbb{Z}_n$:
  - быстро возводить в степень;
  - находить a^{-1};
  - использовать алгоритм Евклида;
  - применять равенство $a^{\Phi(n)} \equiv 1 \pmod{n}$.
- Мы выяснили, что умеем раскладывать n на множители тогда и только тогда, когда умеем вычислять по модулю n квадратные корни.
- И узнали о задаче дискретного логарифма.

Outline

Принцип работы

- У каждого участника — два ключа, секретный и публичный.



RSA: история

- 1973: Clifford Cocks изобрёл RSA, но не опубликовал (работал на разведку).
- 1978: Ron Rivest, Adi Shamir, Leonard Adleman.
- 1983: MIT получил патент на RSA; срок действия истекал в 2003, но в 2000 алгоритм уже был вынесен в public domain.
- Идея — кодировать сообщения так, чтобы раскодирование их опиралось на какую-нибудь сложную вычислительную проблему.

RSA: ключи

- Алгоритм генерации ключей.
  1. Сгенерировать два больших случайных различных простых числа p и q примерно одного размера.
  2. Вычислить $n = pq$ и $\phi = (p-1)(q-1)$.
  3. Выбрать случайное число $1 < e < \phi$, взаимно простое с ϕ.
  4. Алгоритмом Евклида найти $d \equiv e^{-1} \pmod{\phi}$.
  5. Выдать (n, e) как публичный ключ; сохранить d как секретный ключ.

RSA: кодирование и декодирование

- Кодирование (сообщения m с публичным ключом (n, e)).
  1. Представить сообщение как число $0 \leq m \leq n-1$.
  2. Вычислить $c = m^e \pmod{n}$ и выдать c как код.
- Декодирование.
  1. Вычислить $m = c^d \pmod{n}$.
- Почему декодирование работает?

RSA: кодирование и декодирование

- Кодирование (сообщения m с публичным ключом (n, e)).
  1. Представить сообщение как число $0 \leq m \leq n-1$.
  2. Вычислить $c = m^e \pmod{n}$ и выдать c как код.
- Декодирование.
  1. Вычислить $m = c^d \pmod{n}$.
- Почему декодирование работает?
- Потому что

$$c^d \equiv m^{ed} \pmod{n}, \text{ и, т.к. } ed = 1 + k\phi,$$

$$m^{ed} \equiv m^{1+k\Phi} \equiv m \pmod{n}.$$

RSA: простые числа

- Мы тут всё умеем делать, кроме одного: как искать простые числа?
- Если проверять n на простоту, деля на все числа до $\sqrt{n}$, ишак умрёт первым.
- Нам поможет малая теорема Ферма: для простого p и любого $0 < a < p$

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

- Как она может нам помочь?

RSA: простые числа

- Идея: будем выбирать случайные числа $a < n$ и проверять, верно ли, что
$$a^{n-1} \equiv 1 \pmod{n}.$$
- Если нет, то n точно составное. Если да, то n, скорее всего, простое.
- Если n и a выбраны случайно, то вероятность того, что n составное, но $a^{n-1} \equiv 1 \pmod{n}$, около 10^{-13}.
- Но всё-таки бывает; хуже того, бывают *числа Кармайкла* (Carmichael numbers), которые не являются простыми, но при этом для всех a верно $a^{n-1} \equiv 1 \pmod{n}$.
- Придётся чуть модифицировать.

RSA: простые числа

- *Тест Миллера-Рабина*: выразим $n-1=2^b c$, деля пополам.
- Теперь будем брать случайные a и вычислять сначала a^c, а потом последовательно возводить в квадрат.
- Если $a^c \not\equiv 1 \pmod{n}$, но $a^{n-1} \equiv 1 \pmod{n}$, значит, на каком-то шаге мы нашли число, которое не сравнимо с 1, а его квадрат — сравним.
- Если это число не сравнимо с -1, то мы нашли нетривиальный квадратный корень из 1; следовательно, n составное.
- Этот алгоритм для составных n выявляет этот факт с вероятностью $\frac{3}{4}$ (по a).
- Ту же идею можно использовать так, чтобы получился алгоритм вычисления квадратного корня не из 1, а из произвольного числа (упражнение).

RSA: о стойкости

- Во-первых, если уметь раскладывать числа на множители, то RSA решить легко (можно вычислить секретный ключ так же, как при генерации ключей).
- Во-вторых, если враг получит секретный ключ d, он сможет разложить n на множители: т.к. $ed \equiv 1 \pmod{\phi}$, то $ed - 1 = k\phi$, и $a^{ed-1} \equiv 1 \pmod{n}$ для всех a. Выразим $ed - 1 = 2^s t$ для нечётного t.
- Факт: существует $1 \leq i \leq s$, для которого

$$a^{2^{i-1}t} \not\equiv \pm 1 \pmod{n}, \text{ но } a^{2^i t} \equiv 1 \pmod{n}$$

для по крайней мере половины $a \in \mathbb{Z}_n^*$.
- Врагу достаточно выбирать a случайно, искать такой i, и как только найдёт, $\gcd(a^{2^{i-1}t}, n)$ будет нетривиальным делителем n.

RSA: о стойкости

- Иначе говоря, разложение n на множители вычислительно эквивалентно тому, чтобы найти секретный ключ d в криптосистеме RSA.
- Но на самом деле врагу не надо искать d, ему надо раскодировать сообщение.
- *RSA problem*: по данным n, e и c найти такое m, что $m^e \equiv c \pmod{n}$.
- То есть вычислить корень e-й степени по составному модулю n.
- Считается, что эта задача тоже вычислительно сложна, но не известно, эквивалентна ли она разложению n на множители.
- Теперь перейдём к более конкретным проблемам RSA.

RSA: о стойкости

- Какие должны быть простые числа p и q?
- Во-первых, p и q не должны быть слишком маленькими (примерно одной длины в битах), иначе быстрые алгоритмы станут ещё быстрее.
- Но, во-вторых, p и q не должны быть слишком близки друг к другу; иначе можно перебирать числа около $\sqrt{n}$.
- *Сильные простые числа* (strong primes): p — сильное простое, если
  - у $p-1$ есть большой простой делитель r;
  - у $p+1$ есть большой простой делитель;
  - у $r-1$ есть большой простой делитель.

RSA: о стойкости

- Какой выбирать e? Оказывается, что стойкость RSA от e не зависит.
- Поэтому часто выбирают $e = 3$ или $e = 65537$, чтобы легче было кодировать (2 и 17 умножений соответственно).
- Т.е. можно просто выбрать такие p и q, чтобы $(p-1)(q-1)$ не делилось на 3, а потом взять $e = 3$.
- Но при этом есть проблема. Представим, что Алиса послала одно и то же письмо троим друзьям...
- Что здесь будет не так?

RSA: о стойкости

- Пусть Алиса послала три одинаковых письма m трём друзьям с модулями n_1, n_2 и n_3 и публичными ключами $e = 3$.
- Тогда перехвативший сообщения враг знает про сообщение величины $m^3 \pmod{n_1}$, $m^3 \pmod{n_2}$ и $m^3 \pmod{n_3}$.
- Он может по китайской теореме об остатках вычислить $m^3 \pmod{n_1 n_2 n_3}$.
- Но $m^3 < n_1 n_2 n_3$. Значит, он просто получил настоящее m^3, и ему остаётся только взять обычный (не дискретный) кубический корень.
- Значит, сообщения нужно дополнять случайными величинами, чтобы не посылать одно и то же.

RSA: о стойкости

- Ещё проще: по тому же принципу, нельзя выбирать такие m, что $m^e < n$, иначе можно просто подсчитать обычный корень.
- Значит, для $e = 3$ нужно дописывать что-то спереди сообщения.
- Реальные протоколы это и делают.

RSA: о стойкости

- Замечание: во всех алгоритмах с публичным ключом враг может сам кодировать.
- Значит, если он может перебрать все возможные сообщения, он может найти m.
- Поэтому нужно использовать salt (случайное число, дописываемое к сообщению).

RSA: о стойкости

- Далее: представим, что n выбирает какой-нибудь сервер и раздаёт его своим пользователям.
- Это заманчиво, потому что искать хорошие p и q не так легко.
- Почему это не сработает?

RSA: о стойкости

- Далее: представим, что n выбирает какой-нибудь сервер и раздаёт его своим пользователям.
- Это заманчиво, потому что искать хорошие p и q не так легко.
- Почему это не сработает?
- Потому что тогда любой пользователь сможет разложить n на множители и прочесть сообщения всех остальных пользователей. Число n должно быть у каждого своё.

Chosen ciphertext атаки

- Мы говорили об атаках видов ciphertext only, known plaintext и chosen plaintext.
- В криптографии с открытым ключом chosen plaintext доступен всегда, а приходится рассматривать и ещё более страшные атаки.
- Chosen ciphertext: Чарли выбирает несколько шифров и просит Алису их расшифровать. Потом Чарли достаётся кодированное сообщение c, и он пытается его расшифровать сам.
- Adaptive chosen ciphertext: Чарли сначала получает c, а потом может спросить Алису про несколько шифров; единственное ограничение — Чарли не может спрашивать про сам шифр c.

RSA против adaptive chosen ciphertext

- RSA *гомоморфна*, т.е. код сообщения $m_1 m_2$ — это $c_1 c_2$.
- Поэтому против adaptive chosen ciphertext RSA бессильна: получив c, Чарли выбирает случайное число r и спрашивает Алису про $c' = cr^e \pmod{n}$.
- Ответ Алисы — это mr, нужно только разделить на r.
- Как с этим бороться на практике?

RSA против adaptive chosen ciphertext

- RSA *гомоморфна*, т.е. код сообщения $m_1 m_2$ — это $c_1 c_2$.
- Поэтому против adaptive chosen ciphertext RSA бессильна: получив c, Чарли выбирает случайное число r и спрашивает Алису про $c' = cr^e \pmod{n}$.
- Ответ Алисы — это mr, нужно только разделить на r.
- Как с этим бороться на практике?
- Например, можно по запросам всяких там Чарли расшифровывать сообщения только определённого вида, потому что Чарлин c' будет расшифровываться в какую-то случайную строчку; Алиса не должна по запросу расшифровывать нелегитимные сообщения.

Идея

- RSA основана на разложении чисел.
- Но взлом RSA (RSA problem) не эквивалентен разложению n; по крайней мере, мы об этом не знаем.
- Можно ли придумать систему, взлом которой будет *доказуемо* эквивалентен разложению большого числа на простые множители? На чём может быть основана такая система?

Идея

- RSA основана на разложении чисел.
- Но взлом RSA (RSA problem) не эквивалентен разложению n; по крайней мере, мы об этом не знаем.
- Можно ли придумать систему, взлом которой будет *доказуемо* эквивалентен разложению большого числа на простые множители? На чём может быть основана такая система?
- На задаче вычисления квадратного корня.

Криптосистема Рабина: ключи

- Алгоритм порождения ключей.
  1. Сгенерировать два больших простых числа p и q (как и в RSA, лучше сильных).
  2. Вычислить $n = pq$.
  3. Публичный ключ — n, секретный — (p, q).

Криптосистема Рабина: [де]кодирование

- Алгоритм кодирования (вход: публичный ключ n, сообщение m).
  1. Представить m как число, $0 \leq m \leq n-1$.
  2. Вычислить $c = m^2 \pmod{n}$.
- Алгоритм кодирования (вход: секретный ключ (p, q), шифр c).
  1. Вычислить четыре квадратных корня из c (алгоритм был выше).
  2. Как-нибудь из них выбрать.

Криптосистема Рабина: о стойкости

- Задача, стоящая перед пассивным врагом, — это в точности задача вычисления квадратного корня.
- Она эквивалентна разложению n на множители.
- Т.е. мы построили систему, взлом которой *доказуемо* сводится к решению задачи разложения.
- Это криптографы называют *provable security*.
- Криптосистема Рабина подвержена тем же атакам, что RSA, если они упрощают разложение n; поэтому надо выбирать стойкие числа и т.п.

Криптосистема Рабина: о стойкости

- Но против chosen ciphertext дела плохи.
- Враг может использовать криптосистему как оракула в том самом сведении и разложить n. Напомним:
  - враг выбирает случайное число m, подаёт системе m^2;
  - система выдаёт какой-то квадратный корень;
  - поскольку система не знает m, с вероятностью $\frac{1}{2}$ враг получает *другой* квадратный корень и раскладывает n на множители.
- Как с этим справиться?

Криптосистема Рабина: о стойкости

- Можно просто потребовать, чтобы сообщение содержало какие-нибудь особенности.
- Например, определённая часть сообщения должна быть повторена два раза.
- Убиваем двух зайцев. Во-первых, решается проблема с атакой: теперь с подавляющей вероятностью врагу либо ничего не скажут, либо (если он будет подавать правильно оформленные сообщения) скажут его же сообщение.
- Во-вторых, решается проблема того, какой из четырёх корней выбрать Алисе.

Спасибо за внимание!

- Lecture notes и слайды будут появляться на моей homepage:

  `http://logic.pdmi.ras.ru/~sergey/`

- Присылайте любые замечания, решения упражнений, новые численные примеры и прочее по адресам: `sergey@logic.pdmi.ras.ru`, `snikolenko@gmail.com`
- Заходите в ЖЖ  **smartnik**.