

Математические и компьютерные основы защиты информации

Лекция 8



Антон Николаевич Гайдук | УНИВЕР

vk.com/gaidukedu

30 марта 2023 г.

Содержание дисциплины

Раздел I Введение

- Тема 1. Введение. История. Основные понятия.

Раздел II Симметричная криптография

- Тема 2 Классические шифры.
- Тема 3 Поточные алгоритмы шифрования.
- Тема 4 Блочные алгоритмы шифрования.
- Тема 5 Функции хэширования.
- Тема 6 Математические методы криптоанализа.

Раздел III Асимметричная криптография

- Тема 7 Протокол Диффи-Хэллмана.
- Тема 8 Криптосистемы с открытым ключом.
- Тема 9 Электронная цифровая подпись.

Раздел III Асимметричная криптография

Тема 8 Криптосистемы с открытым ключом.

- Модель асимметричной криптосистемы
- Односторонние функции
- Классы P и NP
- Chinese Remainder Theorem
- RSA
- ElGamal

Протокол ДН предлагает интерактивное взаимодействие абонентов. Понятно, что протокол не годится для сценариев типа «электронная почта», когда получатель письма не обязательно может выполнить протокол в момент отправки письма.



- ...каждый может написать письмо...
- ...но только у владельца есть ключ, чтобы открыть ящик...

Асимметричная криптография

Открытый ключ (Public key)



Для зашифрования

Личный ключ (Private Key)



Для расшифрования

Модель асимметричной криптосистемы

- Задание алгоритма генерации (выработки) ключей;
- Задание функции зашифрования;
- Задание функции расшифрования;
- Обоснование работоспособности и надежности.

Модель асимметричной криптосистемы

Асимметричная криптосистема задается шестеркой $\{\mathcal{K}, \mathcal{P}, \mathcal{C}, G, E_{k_e}, D_{k_d}\}$, где

\mathcal{K} — множество ключей (k_e, k_d) , k_e — открытый ключ, k_d — закрытый ключ,

\mathcal{P} — множество открытых текстов,

\mathcal{C} — множество шифртекстов,

G — рандомизированное преобразование выработки ключей $G : \mathbb{N} \rightarrow \mathcal{K}$

E_{k_e} — преобразование зашифрования $E_{k_e} : \mathcal{P} \rightarrow \mathcal{C}$

D_{k_d} — преобразование расшифрования $D_{k_d} : \mathcal{C} \rightarrow \mathcal{P}$

с ограничениями

- однозначность расшифрования: $D_{k_d}(E_{k_e}(p)) = p$ для $\forall p \in \mathcal{P}$.

Основные требования к асимметричной криптосистеме

- Получение пары ключей (k_e, k_d) является легко вычислимой задачей.
- Для каждой пары ключей (k_e, k_d) и каждого открытого текста $p \in \mathcal{P}$ функции E, D легко вычислимы.
- Решение задачи нахождения открытого текста p по шифртексту c и открытому ключу k_e является вычислительно трудной задачей.

Односторонние функции

Отображение $f : X \rightarrow Y$ называется односторонней (однонаправленной) функцией (one-way function), если

- существует полиномиальный алгоритм, для каждого $x \in X$ вычисляющий $f(x)$;
- не существует полиномиального алгоритма, вычисляющего $f^{-1}(y)$ для каждого $y \in Y$.

Если для односторонней функции f существует дополнительный параметр (лазейка) k , зная который $f_k^{-1}(y)$ вычисляется за полиномиальное время, то данная функция f называется односторонней функцией с лазейкой (one-way trap-door function).

Классы P и NP

В теории алгоритмов классом P (от англ. polynomial) называют множество задач разрешимости, для которых существуют алгоритмы решения, время работы которых полиномиально зависит от размера входных данных.

В теории алгоритмов классом NP (от англ. non-deterministic polynomial) называют множество задач разрешимости, решение которых возможно проверить за время, не превосходящее значения некоторого полинома от размера входных данных, при наличии некоторых дополнительных сведений (так называемого сертификата решения).

Задача разложения числа n на множители принадлежит классу NP. Так как предъявленное решение задачи

$$n = p_1^{l_1} p_2^{l_2} \dots p_s^{l_s},$$

можно проверить за полиномиальное время, перемножив числа $p_1^{l_1}, p_2^{l_2}, \dots, p_s^{l_s}$, и сравнив результат с n . Дополнительных данных для доказательства в данном случае не требуется.

$$P = NP ?$$

- Очевидно, что $P \subset NP$
- Верно ли обратное включение $NP \subset P$?
- Данная проблема является одной из семи Задач тысячелетия.

Если положительный ответ на какой-то вопрос можно проверить за полиномиальное время, то правда ли, что ответ на этот вопрос можно также найти за полиномиальное время и используя полиномиальное количество памяти?

Доказательство существования односторонних функций

На данный момент не доказано существование ни одной односторонней функции.

Если $P = NP$; то односторонних функций не существует!

Действительно, для $y \in Y$ существует полиномиальный алгоритм проверки того, что $f(x) = y$. Если $P = NP$, то должен существовать и полиномиальный алгоритм, вычисляющий $f^{-1}(y)$.

Rivest, Shamir, Adleman, 1977

В августе 1977 года в колонке «Математические игры» Мартина Гарднера в журнале Scientific American появилось первое описание криптосистемы RSA. Читателям также было предложено дешифровать английскую фразу, зашифрованную описанной криптосистемой. За расшифровку была обещана награда в 100 долларов США. Фразу расшифровали лишь в 1995 году.

RSA.KeysGeneration()

Вход: $L \in \mathbb{N}$ — длина в битах.

Выход: k_e — public key, k_d — private key.

Шаги:

1. Выбираем два случайных простых числа p и q длиной L бит.
2. Вычисляем: $n = p \cdot q$.
3. Вычисляем: $\varphi(n) = (p - 1) \cdot (q - 1)$.
4. Выбираем случайное число e так, что $1 < e < \varphi(n)$, $\gcd(e, \varphi(n)) = 1$.
5. Вычисляем $d = e^{-1} \pmod{\varphi(n)}$.
6. **Вернуть:** $k_e = (n, e), k_d = (n, d)$.

RSA.Encrypt() и RSA.Decrypt()

$m \in \mathbb{Z}_n$ — открытый текст, $c \in \mathbb{Z}_n$ — шифртекст.

RSA.Encrypt()

Вход: $m, k_e = (e, n)$.

Выход:

$$c = m^e \pmod{n}.$$

RSA.Decrypt()

Вход: $c, k_d = d$.

Выход:

$$m = c^d \pmod{n}.$$

Все три приведенных выше алгоритма являются полиномиальными.

RSA: корректность

Теорема

Пусть $n = pq, p \neq q$ — простые, $e \in \mathbb{N}$, $1 < e < \varphi(n)$, $\gcd(e, \varphi(n)) = 1$. Тогда для каждого $p \in \mathbb{Z}_n$, справедливо равенство

$$D_{k_d}(E_{k_e}(x)) = x.$$

Доказательство. Исходя из определений функций E и D нам необходимо доказать, что

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

Так как $d \equiv e^{-1} \pmod{\varphi(n)} \Rightarrow de \equiv 1 \pmod{\varphi(n)}$, то $ed = r\varphi(n) + 1$.

Пусть $x \not\equiv 0 \pmod{p}$, тогда 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:

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

Следовательно:

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

RSA: пример

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

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

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

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

Зашифрование:

$$m = 5, c = 5^3 \bmod 33 = 26.$$

Расшифрование:

$$26^7 \bmod 33 = 5.$$

Факторизация $n \iff$ вычисление $\varphi(n)$

Теорема

Задача разложения на множители числа $n = pq$ и задача вычисления функции Эйлера $\varphi(n)$ полиномиально эквивалентны.

Доказательство. Если известно разложение числа n на множители p, q то значение функции Эйлера можно вычислить за одну арифметическую операцию $\varphi(n) = (p-1)(q-1)$.

Обратно, если известно $\varphi(n) = (p-1)(q-1)$, то множители p, q удовлетворяют системе

$$\begin{cases} pq = n, \\ (p-1)(q-1) = \varphi(n). \end{cases} \Rightarrow \begin{cases} pq = n, \\ p + q = n + 1 - \varphi(n). \end{cases}$$

По теореме Виета p, q — корни уравнения

$$x^2 - (n + 1 - \varphi(n))x + n = 0.$$

Таким образом

$$p = \frac{n + 1 - \varphi(n) + \sqrt{(n + 1 - \varphi(n))^2 - 4n}}{2}, q = \frac{n}{p}.$$

RSA: стойкость

Теорема

Задача вычисления секретного показателя d полиномиально эквивалентна задаче разложения на множители модуля n .

Чтобы по e вычислить d необходимо знать $\varphi(n)$, т.к. $d \equiv e^{-1} \pmod{\varphi(n)}$, поскольку числа p, q неизвестны атакующему, то возникает задача факторизации большого составного числа.

Задача извлечения корня степени e по модулю n

Зная $e, y = x^e \pmod{n}$, найти x .

Модулярная арифметика

Chinese Remainder Theorem

Теорема

Система сравнений (1) при попарно взаимно простых модулях m_1, \dots, m_k имеет единственное решение x по модулю произведения $m = \prod_{i=1}^k m_i$.

$$\begin{cases} x \equiv a_1 \pmod{m_1}, \\ \dots \\ x \equiv a_n \pmod{m_n}. \end{cases} \quad (1)$$

Chinese Remainder Theorem

Теорема

Система сравнений (2) при попарно взаимно простых модулях p, q имеет единственное решение x по модулю произведения $m = pq$.

$$\begin{cases} x \equiv a \pmod{p}, \\ x \equiv b \pmod{q}. \end{cases} \quad (2)$$

Доказательство.

$$x \equiv aqq_1 + bpp_1 \pmod{pq},$$

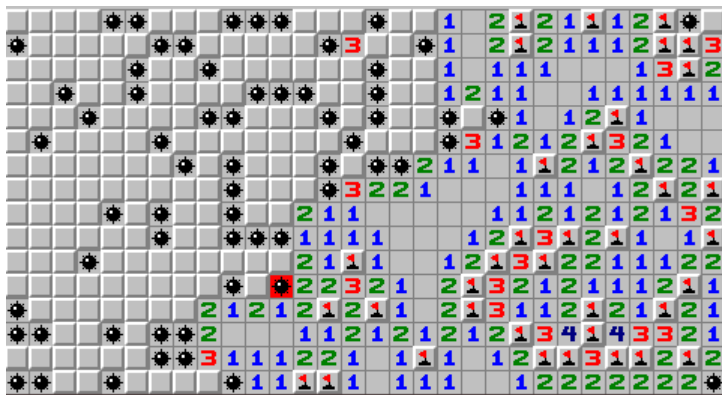
где $p_1 \equiv p^{-1} \pmod{q}$, $q_1 \equiv q^{-1} \pmod{p}$.

RSA-CRT

...In RSA-CRT, it is a common practice to employ the Chinese Remainder Theorem during decryption. It results in a decryption much faster than modular exponentiation. RSA-CRT differs from the standard RSA in key generation and decryption...

Shinde and Fadewar. Faster RSA Algorithm for Decryption Using Chinese Remainder Theorem.
<https://www.techscience.com/icces/v5n4/32277>

RSA in the Wild



RSA in the Wild

 saltstack / salt

 <> Code

 Issues 2,300

 Pull requests 284

 Projects 2

 Wiki



 Security

Change key generation seq

 0.15

 thatch45 authored and basepi committed on May 8, 2013

 Showing 1 changed file with 1 addition and 1 deletion.

▼ 2  salt/crypt.py 

```
@@ -47,7 +47,7 @@ def gen_keys(keydir, keyname, keysize, user=None):
47     priv = '{0}.pem'.format(base)
48     pub = '{0}.pub'.format(base)
49
50 -    gen = RSA.gen_key(keysize, 1, callback=lambda x, y, z: None)
51     cumask = os.umask(191)
52     gen.save_key(priv, None)
53     os.umask(cumask)
```

RSA: выбор параметров

- p и q — большие простые числа (1024-4096 бит),
- число $|p - q|$ должно быть большим (чтобы делители не лежали в окрестности корня из n),
- число $\gcd(p - 1, q - 1)$ должно быть небольшим,
- числа $p \pm 1$, $q \pm 1$ должны содержать большой простой множитель,
- рекомендуется $e = 2^{16} + 1 = 65537$,
- секретный ключ d должен быть большим: доказано, что $d < n^{0.292}$ ведет к снижению стойкости,
- ...

Атаки

1. Математические атаки

- факторизация,
- атака Coppersmith,
- атака на малую открытую экспоненту,
- ...

2. Атаки на реализацию

- padding oracle attacks,
- Timing attacks, power attacks and fault attacks.

Plain RSA attack

Elgamal

ElGamal.KeysGeneration()

Вход: $L \in \mathbb{N}$ — длина в битах.

Выход: k_e — public key, k_d — private key.

Шаги:

1. Выбираем случайное простое p длины L бит.
2. Выбирается целое число g — первообразный корень:
 $\mathbb{Z}_p^* = \{1, g, g^2, \dots, g^{p-2}\}.$
3. Выбирается случайно x такое, что $1 < x < p - 1$.
4. Вычисляется $y = g^x \pmod{p}$.
5. **Открытый ключ:** $k_e = (p, g, y)$.
6. **Личный ключ:** $k_d = (x)$.

ElGamal.Encrypt() и ElGamal.Decrypt()

$m \in \mathbb{Z}_p^*$ — открытый текст, $c \in \mathbb{Z}_p^* \times \mathbb{Z}_p^*$ — шифртекст.

ElGamal.Encrypt()

Вход: $m, (p, g, y)$.

Выход: c

Шаги:

1. $k \xleftarrow{R} \{2, \dots, p-2\}$.
2. вычисляются $a = g^k, b = y^k m$.
3. вернуть $c = (a, b)$.

ElGamal.Decrypt()

Вход: $c = (a, b), x$.

Выход:

$$m = b(a^x)^{-1}.$$

Очевидным преимуществом криптосистемы Эль-Гамала является ее вероятностный характер, что обусловлено случайным выбором ключа k . Недостатком является удвоение длины зашифрованного текста по сравнению с начальным текстом. Кроме того, для зашифрования различных сообщений необходимо использовать разные ключи. Можно показать, что если сообщения m_1 и m_2 зашифрованы с помощью одного и того же ключа k , то для соответствующих шифртекстов (a_1, b_1) и (a_2, b_2) выполняется соотношение $b_1 b_2^{-1} = m_1 m_2^{-1}$. Следовательно, если известно m_1 , то легко находится m_2 .

