Алгоритмы и структуры данных

Лекция 11. Алгоритмы шифрования.

(c) Глухих Михаил Игоревич, glukhikh@mail.ru

Вступление:
простые алгоритмы шифрования

Вступление: простые алгоритмы шифрования

- Перестановка букв в сообщении (транспозиция)
- «Пример маршрутной перестановки» → «ешоеомрнрниатеаирмуптккпррйсв»

п	р	и	м	е
р	м	а	р	ш
р	у	т	н	о
й	п	е	р	е
с	т	а	н	о
в	к	и		

Вступление: простые алгоритмы шифрования

- Замена букв в сообщении по таблице (простая замена)
- ROT13: “HELLO” → “URYYB”
  - A B C D E F G H I J K L M
  - N O P Q R S T U V W X Y Z

Вступление: простые алгоритмы шифрования

- Перестановка букв в сообщении (транспозиция)
- Замена букв в сообщении по таблице (простая замена)
- XOR-шифрование (и близкие к нему)

Цели шифрования

- Конфиденциальность
  - Информация доступна только авторизованным пользователям
- Целостность
  - Предотвращение изменения информации
- Идентифицируемость
  - Проверка отправителя информации

Алгоритмы шифрования

- По открытости ключа
  - С закрытым ключом
  - С открытым ключом
- По симметричности (ключ шифрации = ключ дешифрации)
  - Симметричное
  - Асимметричное
- По цели
  - Шифрация
  - Дешифрация
  - Электронная подпись

Шифрование с открытым ключом [и закрытым]

- Открытый ключ: известен кому угодно, т.е. его не требуется скрывать
  - При этом предполагается, что у получателя сообщения есть ещё закрытый ключ для расшифровки

Шифрование с открытым ключом [и закрытым]

- Открытый ключ: известен кому угодно, т.е. его не требуется скрывать
  - При этом предполагается, что у получателя сообщения есть ещё закрытый ключ для расшифровки
- Закрытый ключ: известен **только** получателю

Шифрование с одним секретным (aka закрытым) ключом

- Секретный ключ: известен только двоим участникам передачи (в этом случае возникает отдельный сложный вопрос о передаче секретного ключа – часто для этого используется алгоритм с открытым ключом)

Шифрование с открытым ключом: принцип

- У получателя есть закрытый ключ (он его придумал или сгенерировал, и никому не показывает)
- Из него он (известным преобразованием) получает открытый ключ и публикует его
  - Преобразование имеет такой характер, что обратное преобразование (открытый ключ → закрытый ключ) намного более трудоёмко (NP-полная задача или близкая к ней)

Шифрование с открытым ключом: принцип

- У получателя есть закрытый ключ (он его придумал или сгенерировал, и никому не показывает)
- Из него он (известным преобразованием) получает открытый ключ и публикует его
  - Преобразование имеет такой характер, что обратное преобразование (открытый ключ → закрытый ключ) намного более трудоёмко (NP-полная задача или близкая к ней)
- Далее, открытый ключ используется для шифрации…
- … а закрытый – для дешифрации

Электронная подпись

- Выполняется в обратном порядке
- Автор документа с помощью закрытого ключа формирует электронную подпись (например, шифруя часть информации в документе)
- Кто угодно другой может проверить его подпись с помощью открытого ключа (расшифровывая и сравнивая с оригиналом)

Шифрование с открытым ключом: RSA

- Алгоритм разработан в 1977 году
  - = Rivest + Shamir + Adleman (три фамилии)
- Базовые функции
  - Открытый → Закрытый ключ: факторизация большого числа
  - Закрытый → Открытый ключ: произведение двух простых чисел
  - Шифрация: возведение в степень по модулю большого числа
  - Дешифрация: вычисление функции Эйлера
    - = количеству натуральных чисел, меньших N и взаимно простых с N

RSA: генерация ключей

- Берём два простых числа (128, 256, 512, 1024, 2048 бит…)
- Считаем их произведение: N = P x Q
- Считаем функцию Эйлера Э(N) = (P-1) x (Q-1)
- Выбираем E, простое или взаимно простое с Э(N) – например, 17, 257, **65537**
- Вычисляем D: (E x D) % Э(N) = 1
  - Обычно для этого используется расширенный алгоритм Евклида
    - E x **D** + Э(N) x **K** = **GCD(E, Э(N))** = 1
- Public key = E, N
- Private key = D

RSA: шифрование

- NB: задача шифрования чего угодно может быть сведена к задаче шифрации чисел!

RSA: шифрование

- Сообщение = число в интервале 0…N-1
- Берём исходное сообщение M
- Формируем $C(M) = M^E \% N$

RSA: дешифрование

- Берём принятое C
- Говорим, что $M = U(C) = C^D \% N$

Применение

- Чаще всего с помощью RSA шифруется только ключ для симметричного шифрования
- После чего уже с его помощью передаётся основной текст

Шифрование с закрытым ключом: AES

- AES = Advanced Encryption Standard
- Появление: 2000-2001
- Симметричное шифрование

AES: порядок шифрования

- Выбираем ключ (128, 192 или 256 бит)
- Делим входные данные на блоки по 128 бит (16 байт), каждый из блоков преобразуется в матрицу 4x4 байта
- Выполняется 10, 12 или 14 раундов преобразований матрицы:
  - Перестановка ячеек по фиксированной таблице
  - Сдвиг строк влево
  - Перемножение колонок на фиксированный многочлен
  - XOR для каждой ячейки

AES: порядок дешифрации

- Обратен шифрации

Шифрование: Java

- ГСЧ: java.security.SecureRandom (extends Random)
  - Криптографически стойкий (почти ☺)
    - … “Minimally complies with the statistical random number generator tests”
    - «Тест на следующий бит» -- невозможно, зная первые N бит, предсказать (N+1)-й с вероятностью >50%, используя полиномиальный алгоритм

Шифрование: Java

- Генератор пар ключей: java.security.KeyPairGenerator
  - getInstance(algorithm)
    - RSA, DSA (= Digital Signature Algorithm), EC (= Elliptic Curve), DiffieHelman
  - initialize(keysize, [secureRandom])
  - generateKeyPair()

Шифрование: Java

- Генератор ключей: java.security.KeyGenerator
  - getInstance(algorithm)
    - AES, DES (= Data Encryption Standard), …
  - init(keysize, [secureRandom])
  - generateKey()

Шифрование: Java

- Пара ключей: java.security.KeyPair
  - = publicKey + privateKey
- Один ключ: java.security.Key
  - PublicKey
  - PrivateKey
  - SecretKey

Шифрование: Java

- Хранилище ключей: java.security.KeyStore
  - getInstance(KeyStore.getDefaultType())
  - load(inputStream, password)
  - KeyStore.PasswordProtection(password)
  - getEntry(name, protectionParameter)
    - Так достаётся закрытый или секретный ключ
  - getCertificate(name)
    - Из сертификата берётся открытый ключ

Шифрование: Java

- Сертификат: java.security.cert.Certificate
  - «Сертификат открытого ключа» =
    Открытый ключ + информация о владельце + цифровая подпись
  - Используется:
    - Для проверки цифровой подписи
      - verify(publicKey) throws exception on incorrect private key used
    - Для шифрования
      - getPublicKey()

Шифрование: Java

- Цифровая подпись: java.security.Signature
  - getInstance(algorithm)
    - SHA1withRSA, SHA1withDSA
    - SHA1 = Secure Hash Algorithm,
      алгоритм криптографического хэширования
  - Инициализация или/или
    - initSign(privateKey)
    - initVerify(publicKey or certificate)
  - update(byte[] inputBytes) + byte[] sign()
  - verity(byte[] signatureBytes)

Шифрование: Java

- Шифровщик/дешифровщик: javax.crypto.Cipher
  - getInstance(transformation)
    - RSA, AES, …
  - init(mode, key)
    - ENCRYPT_MODE, DECRYPT_MODE
  - byte[] doFinal(byte[] input)

Шифрование: Java

- Кодировщик двоичных данных: java.util.Base64
  - Стандарт кодирования байт с помощью 64 символов (A-Z, a-z, 0-9 и ещё два символа)
  - 3 исходных байта (3x8 бит) → 4 выходных символа (4x6 бит), по таблице соответствия
  - String Base64.getEncoder().encodeToString(inputBytes)
    - byte[] inputString.getBytes(charset)
  - byte[] Base64.getDecoder().decode(inputString)
    - String(inputBytes, charset)

Спасибо за внимание!

- Пример RSA-шифрования на Java: https://gist.github.com/nielsutrecht/855f3bef0cf559d8d23e94e2aecd4ede