TIA5302 – Teknologi Blockchain

HO 06 - Asymmetric Key Cryptography

Opim Salim Sitompul

Department of Computer Science
Universitas Sumatera Utara

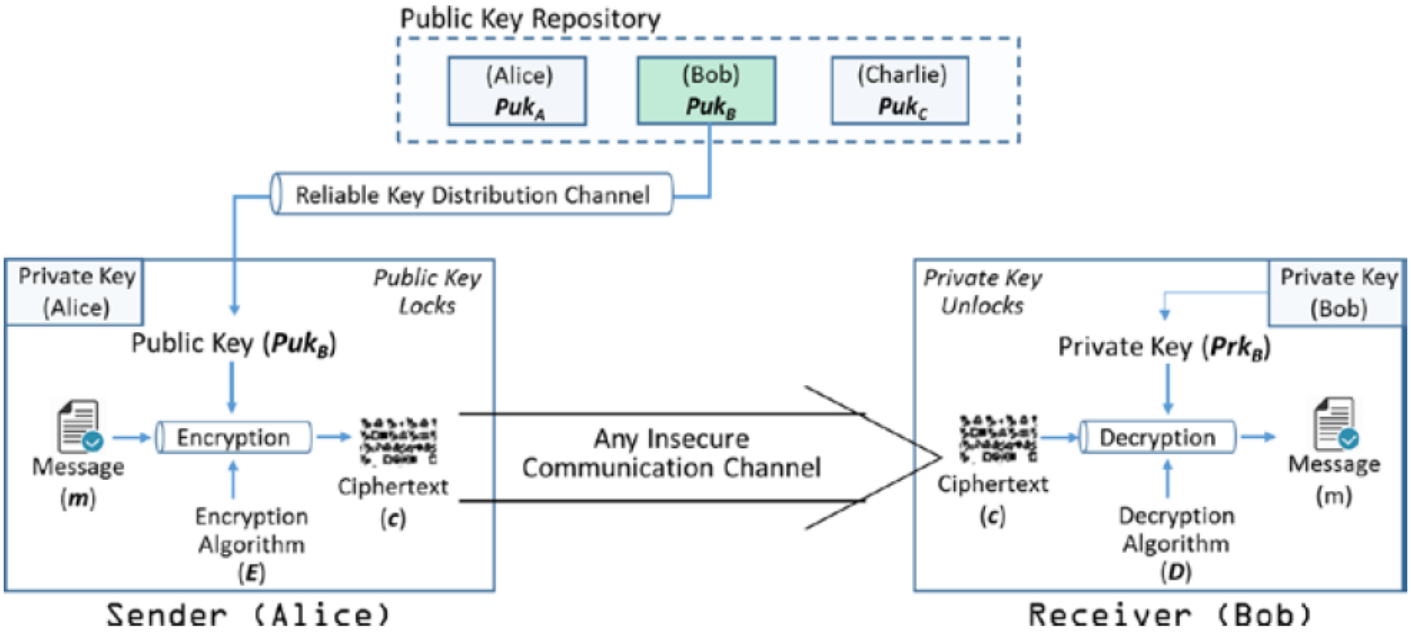
2023

Outline

Pengantar

- Dikenal juga dengan nama *public key cryptography*, diperkenalkan oleh Diffie and Hellman [1].
- Menyelesaikan masalah distribusi key dalam sistem symmetric cryptography dengan memperkenalkan *digital signatures*.
- Asymmetric key cryptography tidak menghilangkan kebutuhan terhadao symmetric key cryptography, melainkan saling melengkapi.

Mengenal Asymmetric Key Cryptography



Gambar 1: Asymmetric cryptography for confidentiality [2]

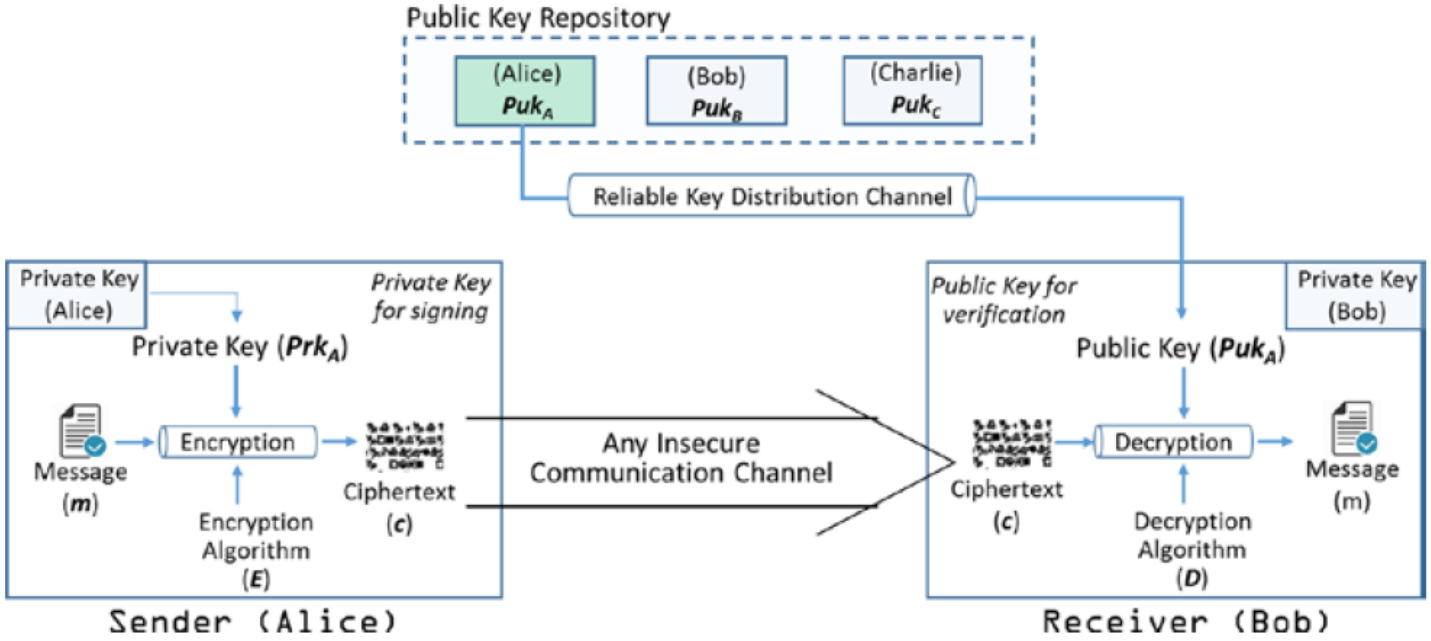
Konfidensialiti dan Otentikasi

- Konfidensialiti (*Confidentiality*):
  - Proses menyembunyikan makna sebuah pesan dengan cara penyandian (*encoding*).
  - Memastikan bahwa pesan terenkripsi tidak mungkin dipahami tanpa mengetahui *secret key*.
- Otentikasi (*Authentication*):
  - Proses membuktikan siapa seseorang itu.
  - Memastikan bahwa suatu data berasal dari dimana data itu dinyatakan berasal.

Konfidensialiti dan Otentikasi

- Alice—The Sender:
  - Encrypt the plaintext message m using encryption algorithm **E** and the public key Puk_{Bob} to prepare the ciphertext c.
  - $c = E(Puk_{Bob}, m)$
  - Send the ciphertext c to Bob.
- Bob—The Receiver:
  - Decrypt the ciphertext c using decryption algorithm **D**
  - $m = D(Prk_{Bob}, c)$

Konfidensialiti dan Otentikasi



Gambar 2: Asymmetric cryptography for authentication [2]

Konfidensialiti dan Otentikasi

- Pesan disiapkan menggunakan private key Alice, sehingga bisa dipastikan hanya berasal dari Alice, sehingga keseluruhan pesan bertindak sebagai *digital signature*.
- Confidentiality dan authentication keduanya diperlukan, untuk ini public key encryption harus dilakukan dua kali.
  - Pesan pertama sekali dienkripsi menggunakan *private key* pengirim (*sender*) untuk memberikan *digital signature*.
  - Kemudian dienkripsi menggunakan *public key* penerima (*receiver*) untuk memberikan *confidentiality*.
    $c = E[Puk_{Bob}, E(Prk_{Alice}, m)]$
    $m = D[Puk_{Alice}, D(Prk_{Bob}, c)]$

Konfidensialiti dan Otentikasi

- *Public key* hendaklah disimpan di sebuah repositori publik yang dapat diakses setiap orang.
- *Private key* hendaklah disimpan di tempat rahasia yang terjaga baik.
- Public key cryptography juga memberikan suatu cara otentifikasi.
- Si penerima Bob, dapat memverifikasi keaslian asal pesan m dengan cara yang sama.

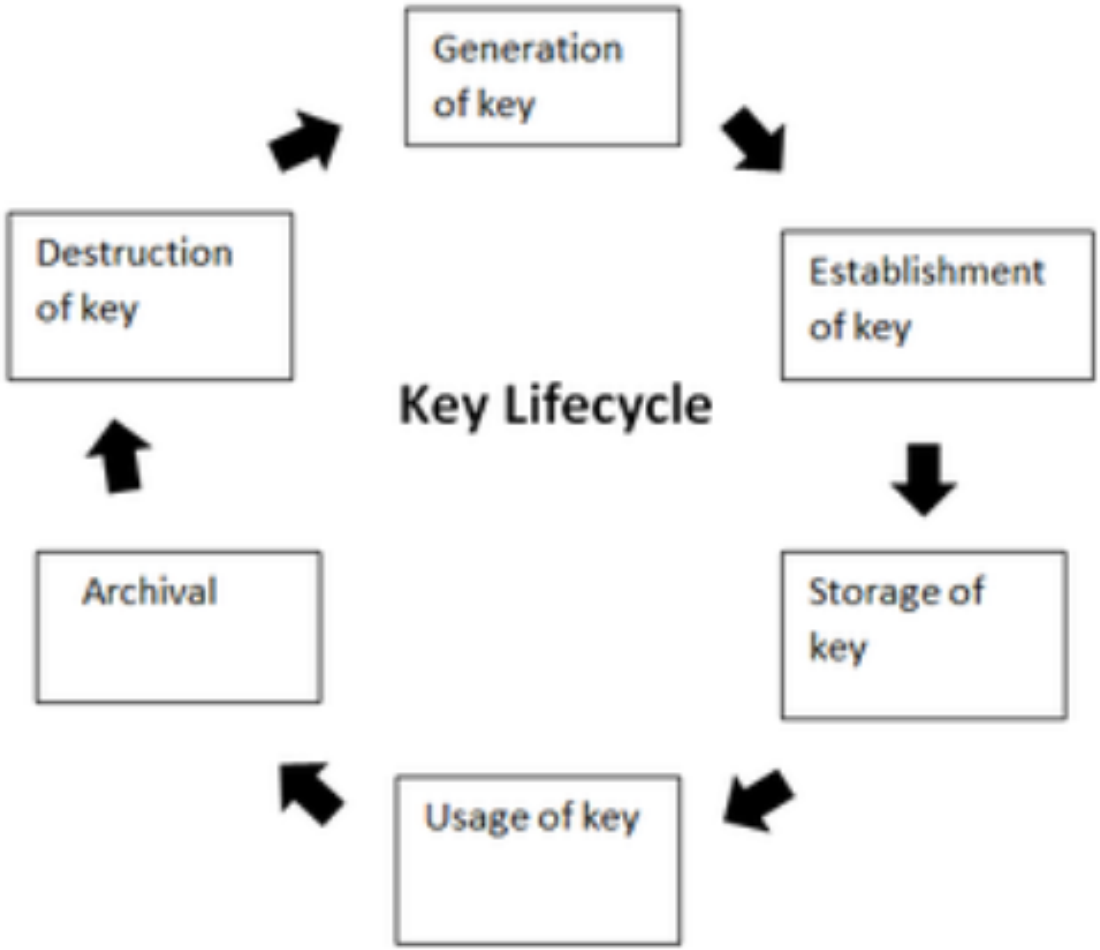
Public key infrastructure

- *Public key infrastructure* (PKI) adalah sebuah sistem hardware, software, manusia, kebijakan, dan prosedur yang menciptakan, mengelola, menyimpan, mendistribusikan, dan menarik kembali sertifikat digital (*digital certificate*) berdasarkan asymmetric cryptography.
- Sertifikat digital adalah kredensial aman yang memungkinkan otentikasi, enkripsi, dan proteksi data bagi pengguna (internet), organisasi dan perangkat yang terkoneksi.

Public key infrastructure

- PKI dipasang di dalam *web browser* dan mendukung lalu lintas internet publik demikian pula komunikasi internal dan jaringan.
- PKI menggunakan sepasang kunci: public key dan private key, yang secara matematis berhubungan tetapi tidak identik, untuk melakukan fungsi-fungsi kriptografi.
- Mengelola Key dalam Cryptosystem:
  - Sekuriti sebuah cryptosystem bergantung kepada kunci-kunci yang digunakan.
  - Cryptographic key adalah potongan data yang harus dikelola dengan administrasi yang aman meliputi pengelolaan *key life cycle*.

Public key infrastructure



Gambar 3: Key life cycle (Sumber: public-key-infrastructure)

Public key infrastructure

- Bagaimana memastikan bahwa *public key* yang digunakan untuk mengenkripsi pesan adalah benar-benar *public key* dari resipien yang dimaksudkan dan bukan *intruder* atau *eavesdropper*?
- Penyelesaiannya adalah dengan menggunakan pihak ketiga yang dipercaya, disebut *public key infrastructure* (PKI).
  - Melalui PKI, keaslian *public key* dijamin oleh proses pengesahan atau notarisasi (pencatatan notaris) terhadap identitas pengguna.
  - PKI memberikan *public key* terverifikasi dengan menyematkannya (*embed*) dalam sebuah *security certificate* melalui penandatanganan digital (*digital signing*).

Cara Kerja Public key infrastructure

- PKI and Encryption:
  - Root PKI melibatkan penggunaan kriptografi dan teknik-teknik enkripsi, baik enkripsi symmetric maupun asymmetric menggunakan sebuah public key.
- Certifying Authorities:
  - CA mengeluarkan dan memverifikasi sertifikat.
  - Otoritas ini memastikan bahwa informasi dalam sertifikat adalah *real* dan benar serta sertifikat ditandatangani secara digital.
  - Peranan CA adalah seperti digambarkan pada Gambar 3.

Cara Kerja Public key infrastructure

- Public Key Certificate (Digital Certificate):
  - Diberikan ke masyarakat dan sistem elektronik untuk mengidentifikasi ketunggalannya dalam dunia digital, menggunakan ITU standard X.509.
    - Certification Authority (CA) menyimpan public key seseorang bersama dengan informasi lain tentang si klien di dalam digital certificate.
    - Informasi itu ditandatangani dan digital signature juga disimpan dalam certificate.
    - Afirmasi untuk public key kemudian dikeluarkan dengan memvalidasi tandatangan menggunakan public key dari Certification Authority.

Proses Pembuatan Sertifikat

- Membuat private dan public key.
- CA meminta atribut identifikasi pemilik private key.
- Public key dan disandikan ke dalam Certificate Signing Request (CSR).
- Pemilik key menandatangani CSR untuk mebuktikan kepemilikan private key.
- CA menandatangani sertifikat setelah validasi.

RSA

- Diambil dari nama-nama Ron Rivest, Adi Shamir, dan Leonard Adleman.
- Berdasarkan sulitnya secara praktis memfaktorkan bilangan yang sangat besar.
- Dalam RSA, plaintext dan ciphertext adalah bilangan-bilangan bulat (*integer*) antara 0 dan *n - 1* untuk suatu *n* tertentu.

RSA

- Prinsip dasar RSA berasal dari pengamatan praktis bahwa mencari tiga bilangan integer positip sangat besar e, d, dan n, sedemikian hingga dengan eksponensiasi modular untuk semua integer m $(0 \leq m < n)$:

$$(m^e)^d \equiv m \pmod{n}$$

dan dengan mengetahui e dan n, atau bahkan m, akan sangat sulit mencari d.

Modular Arithmetic

- Misalkan *m* berupa sebuah bilangan bulat positip disebut modulus.
- Dua bilangan bulat *a* dan *b* adalah *congruent modulo m* apabila:
  - a $\equiv$ b (mod m), menghasilkan $a - b = m \cdot k$ untuk beberapa bilangan bulat *k*.
- Contoh:
  - jika $a \equiv 16$ (*mod* 10) maka *a* dapat memiliki penyelesaian berikut: *a* = ..., -24, -14, -4, 6, 16, 26, 36, 46
  - Tiap-tiap bilangan ini dikurangkan dengan 16 adalah habis dibagi 10.
  - Contoh,
    $-24 - 16 = -40$, habis dibagi 10.
  - Catatan: $a \equiv 36$ (*mod* 10) juga dapat memberikan hasil yang sama untuk *a*.

Modular Arithmetic

- Quotient-Remainder theorem menyatakan bahwa hanya ada sebuah penyelesaian tunggal a yang memenuhi syarat $0 \leq a < m$.
- Dalam contoh sebelumnya $a \equiv 16 \ (mod\ 10)$, hanya nilai 6 yang memenuhi syarat $0 \leq 6 < 10$.
- Syarat inilah yang akan digunakan dalam proses enkripsi/dekripsi algoritma RSA.

Modular Arithmetic

- Inverse Modulus:
  - Jika b adalah inverse pada a modulo m, maka dapat disajikan sebagai:
    $a\,b \equiv 1\ (mod\ m)$, yang mengakibatkan bahwa $a\,b - 1 = m \cdot k$ untuk beberapa bilangan bulat k.
  - Contoh:
    3 memiliki inverse 7 modulo 10 karena
    $3 \cdot 7 = 1\ (mod\ 10) \Rightarrow 21 - 1 = 20$, yang habis dibagi 10.

Pembentukan Pasangan Key

- Pasangan private key dan public key diperlukan untuk setiap pihak yang berpartisipasi dalam *asymmetric crypto-communication*.
- Dalam skema RSA, public key terdiri dari (e, n) dimana n disebut *modulus* dan e disebut *public exponent*.
- Begitu pula, private key terdiri dari (d, n), dimana n adalah *modulus* dan d adalah *private exponent*.

Pembentukan Pasangan Key

- Contoh pembentukan key:
  - Buat pasangan dari dua bilangan prima besar p dan q.
    - $\Rightarrow$ Untuk contoh hanya diambil dua bilangan prima p = 7 dan q = 17.
  - Hitung RSA modulus (n) sebagai $n = pq$.
    - $\Rightarrow$ n hendaklah sebuah bilangan yang besar, tipikalnya minimum 512 bit.
    - $\Rightarrow$ Dalam contoh, modulus $(n) = pq = 119$.
  - Cari sebuah *public exponent* e sedemikian hingga $1 < e < (p-1)(q-1)$ dan tidak boleh ada faktor yang sama untuk e dan $(p-1)(q-1)$ kecuali 1.
    - $\Rightarrow$ Berarti bahwa e dan $(p-1)(q-1)$ adalah *coprime*.
    - $\Rightarrow$ Catatan: Boleh jadi ada beberapa nilai yang memenuhi syarat ini dan diambil sebagai e, tetapi hendaklah diambil salah satunya.

Pembentukan Pasangan Key

- Pada contoh, $(p-1)\,(q-1) = 6 \times 16 = 96$.
- Sehingga e akan memiliki prima relatif ≤ 96.
- Misalkan $e = 5$.
  - Pasangan bilangan (e, n) membentuk *public key* dan hendaklah dibuat *public*.
  - Sehingga, pada contoh ini public key adalah (5, 119).

Pembentukan Pasangan Key

- Hitung *private exponent d* menggunakan *p*, *q*, dan *e* dengan mempertimbangkan bilangan *d* adalah inverse dari *e* modulo (p - 1) (q - 1).
- Akibatnya, *d* apabila dikalikan dengan *e* adalah sama dengan 1 modulo (p - 1) (q - 1) dan d < (p - 1) (q - 1).
- Disajikan sebagai:

$$e\ d = 1\ mod\ (p-1)\ (q-1) \tag{1}$$

- Catatan: meskipun tidak diturunkan secara langsung, terdapat hubungan antara *private key* dan *public key*.

Pembentukan Pasangan Key

- Pada contoh, d harus dicari sedemikian hingga memenuhi persamaan (1).
- Berarti, $5\ d = 1 \bmod 96$ dan $d < 96$.
- Dengan menyelesaikan untuk beberapa nilai d (dapat dihitung menggunakan perluasan versi algoritma Euclid), terlihatlah bahwa $d = 77$ memenuhi syarat.
- Perhatikan matematika berikut:

  $77 \times 5 = 385$ dan $385 - 1 = 384$ adalah habis dibagi oleh 96 karena $4 \times 96 + 1 = 385$
- Dengan demikian diperoleh *private key* (77, 119).

Algotma Euclid adalah sebuah prosedur untuk mencari greatest common divisor (GCD) dari dua bilangan, yang merupakan bilangan terbesar yang habis membagi keduanya tanpa sisa.

Coprime

- Definisi *coprime*:
  - Dua atau lebih bilangan bulat positip yang tidak memiliki faktor bilangan positip yang sama kecuali 1.
  - Bermakna bahwa *greatest common divisor* (GCD) dari bilangan-bilangan coprime adalah 1.
  - Contoh: 21 dan 22 adalah *coprime*.
    Faktor-faktor dari 21 adalah 1, 3, 7 dan 21.
    Faktor-faktor dari 22 adalah 1, 2, 11 dan 22.
    Faktor yang sama hanyalah 1.
- Berikut adalah pasangan bilangan coprime antara 10 s/d 20:

  $[[10, 11], [10, 13], [10, 17], [10, 19], [11, 12],$
  $[11, 13], [11, 14], [11, 15], [11, 16], [11, 17], [11, 18],$
  $[12, 13], [12, 17], [13, 14], [13, 15], [13, 16], [14, 15]]$

Enkripsi/Dekripsi Menggunakan Pasangan Key

- Mengenkripsi pesan plaintext m untuk memperoleh pesan ciphertext c:
  - $c = m^e \ (mod\ n)$ dengan public key (e, n) dan pesan plaintext m.
- Mendekripsi pesan ciphertext c untuk mendapatkan pesan plaintext m:
  - $m = c^d \ (mod\ n)$ dengan private key (d, n) dan pesan ciphertext c.

Enkripsi/Dekripsi Menggunakan Pasangan Key

- Skema RSA adalah block cipher dimana input dibagi ke dalam blok-blok kecil yang dapat digunakan oleh algoritma RSA.
- Begitu pula dengan plaintext dan ciphertext yang merupakan bilangan bulat dari 0 hingga $n - 1$ untuk beberapa bilangan bulat n yang diketahui oleh sender dan receiver.
- Hal ini berarti input plaintext disajikan sebagai integer, dan ketika diberikan ke RSA untuk diubah menjadi ciphertext, keduanya adalah integer, tetapi tidak sama sebagai input (di-enkripsi)

Enkripsi/Dekripsi Menggunakan Pasangan Key

- Misalkan sender ingin mengirim sebuah pesan text ke receiver dengan public key (e, n).
- Sender memecah pesan text menjadi blok-blok yang dapat disajikan sebagai deretan bilangan yang lebih kecil daripada n.
- Ekivalensi ciphertext terhadap plaintext dapat dicari menggunakan $c = m^e \ (mod\ n)$.
  - Jika plaintext (m) adalah 19 dan public key adalah (5, 119) dengan $e = 5$ dan $n = 119$,
  - Maka ciphertext c adalah
    $19^5 \ (mod\ 119) = 2,476,099 \ (mod\ 119) = 66$,
    yang merupakan remainder, dan 20,807 adalah hasil bagi yang tidak digunakan.
  - Sehingga, $c = 66$.

Enkripsi/Dekripsi Menggunakan Pasangan Key

- Apabila ciphertext 66 diterima di pihak receiver, maka untuk mendapatkan plaintext perlu didekripsi menggunakan $m = c^d \ (mod\ n)$.
- Receiver telah memiliki private key (d, n) dengan $d = 77$ dan $n = 119$, dan menerima ciphertext $c = 66$ oleh sender.
- Sehingga, receiver dapat dengan mudah memperoleh kembali plaintext menggunakan nilai-nilai ini $m = 66^{77} \ (mod\ 119) = 19$ (menggunakan Python, pow(66,77,119)).

Langkah-langkah Algoritma RSA

- Algoritma RSA terdiri dari empat langkah:
  1. Pembuatan key
  2. Pendistribusian key
  3. Enkripsi
  4. Dekripsi

Pembuatan Key

- Membuat Public Key
  - Pilih dua buah bilangan prima. Misalkan P = 53 dan Q = 59.
  - Bagian pertama dari Public key: n = P*Q = 3127.
  - Ambil sebuah eksponen kecil *e*
    - Berupa bilangan integer.
    - Bukan faktor dari $\phi(n)$.
    - 1 < e < $\phi(n)$
    - Misalkan $e = 3$.
  - Jadi public key = (3, 3127)

Pembuatan Key

- Membuat Private Key
  - Hitung $\phi(n)$, sehingga $\phi(n) = (P-1)(Q-1)$
  - Diperoleh $\phi(n) = 3016$
- Hitung private key, d:
  - $d = (k * \phi(n) + 1)/e$ untuk beberapa integer k
  - Untuk $k = 2$, nilai d adalah 2011.
- Diperoleh Public Key (e, n) = (3, 3127) dan Private Key(d, n) = (2011, 3127)

Distribusi Key

- Andaikan Bob ingin mengirim informasi ke Alice.
- Dengan menggunakan RSA, Bob harus mengetahui public key Alice untuk mengenkripsi pesan.
- Alice harus menggunakan private key nya untuk mendekripsi pesan.
- Untuk memungkinkan Bob mengirim pesan ter-enkripsinya, Alice mentransmisikan publi key nya (n, e) ke Bob via rute yang handal, tetapi tidak perlu rahasia.
- Private key Alice tidak pernah didistribusikan.

Enkripsi

- Lakukan enkripsi "HI"
  - H = 8, I = 9
  - Data enkripsi $c = (89^e)\ mod\ n$
    $c = (89)^3 mod\ 3127 = 1394$

Dekripsi

- Lakukan dekripsi 1394
  - Data dekripsi $d = (c^d)\ mod\ n$
    $d = (1394)^{2011}\ mod\ 3127 = 89$
    8 = H and I = 9, yaitu "HI".

Implementasi Algoritma RSA I

```
1  import random
2
3  def is_prime(num):
4    if num <= 1:
5      return False
6    for i in range(2, int(num**0.5) + 1):
7      if num % i == 0:
8        return False
9      return True
10
11 def gcd(a, b):
12   while b != 0:
13     a, b = b, a % b
14   return a
15
```

Implementasi Algoritma RSA II

```
16 def mod_inverse(a, m):
17     m0, x0, x1 = m, 0, 1
18     while a > 1:
19         q = a // m
20         m, a = a % m, m
21         x0, x1 = x1 - q * x0, x0
22         return x1 + m0 if x1 < 0 else x1
23
24 def generate_keypair(bits):
25     p, q = random_prime(bits), random_prime(bits)
26     n = p * q
27     phi = (p - 1) * (q - 1)
28     e = random.randrange(1, phi)
29     while gcd(e, phi) != 1:
30         e = random.randrange(1, phi)
```

Implementasi Algoritma RSA III

```
31    d = mod_inverse(e, phi)
32    return ((e, n), (d, n))
33
34  def random_prime(bits):
35    while True:
36      num = random.getrandbits(bits)
37      if is_prime(num):
38        return num
39
40  def encrypt(message, public_key):
41    e, n = public_key
42    cipher_text = [pow(ord(char), e, n) for char in
          message]
43    return cipher_text
44
```

Implementasi Algoritma RSA IV

```
45 def decrypt(cipher_text, private_key):
46     d, n = private_key
47     plain_text = ''.join([chr(pow(int(char), d, n))
           for char in cipher_text])
48     return plain_text
49
50 # Example usage
51 if __name__ == "__main__":
52     bits = 10 # the number of bits for the key size
53     public_key, private_key=generate_keypair(bits)
54     print("Public key: ", public_key)
55     print("Private key: ", private_key)
56
57     message = "Hello, RSA!"
58     print("Original message:", message)
```

Implementasi Algoritma RSA V

```
59
60    encrypted_message = encrypt(message, public_key
          )
61    print("Encrypted message:", ''.join([str(i) for
          i in encrypted_message]))
62
63    decrypted_message = decrypt(encrypted_message,
          private_key)
64    print("Decrypted message:", decrypted_message)
```

Dekripsi

- Output:

  Public key: (22079, 211031)

  Private key: (119519, 211031)

  Original message: Hello, RSA!

  Encrypted message:

  109573127010144891144891928537819218212019728815677918300465875

  Decrypted message: Hello, RSA!

References I

- New Directions in Cryptography, Whitfield Diffie & Martin E. Hellman, IEEE Transactions On Information Theory, Vol. It-22(6): 644-654, 1976.
- A Beginner's Guide to Building Blockchain Solutions, Bikramaditya Singhal, Gautam Dhameja, Priyansu Sekhar Panda, Appress, New York, USA, 2018.