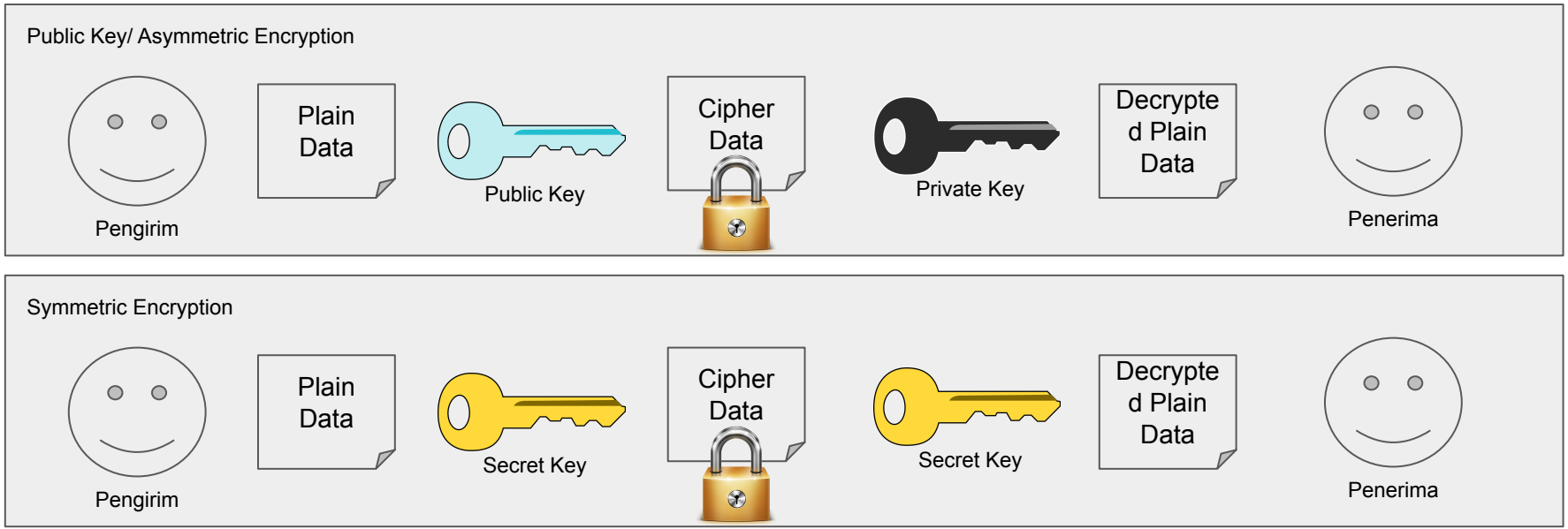
Memahami cara kerja RSA Cryptography Algorithm

http://wuriyan.to

RSA Algorithm

RSA adalah salah satu algoritma kriptografi dengan sistem Public Key Encryption (varian lain: Elliptic Curve, Diffie Hellman). Yang artinya, untuk melakukan enkripsi, sistem enkripsi ini menggunakan Public Key/ Kunci Publik. Sedangkan untuk proses dekripsi, sistem enkripsi ini menggunakan Private Key/ Kunci Privat. Sesuai namanya, Public Key bisa anda bagikan ke siapapun yang ingin mengirim pesan kepada anda. Sedangkan Private Key harus anda jaga baik-baik, dan tidak boleh jatuh ke tangan siapapun. Dengan penggunaan dua kunci, yaitu Private Key dan Public Key. Sistem enkripsi ini juga disebut Asymmetric Encryption. Berlawanan dengan Symmetric Encryption seperti AES, yang menggunakan kunci yang sama untuk melakukan enkripsi dan dekripsi.



Mathematical Term Penting untuk memahami RSA Algorithm

Factor

Faktor dalam Matematika adalah angka yang bisa membagi angka lain tanpa sisa/remainder.

Contoh:

Faktor dari 10 adalah 1, 2, 5, 10

Karena

10 / 1 = 10 (tanpa sisa/remainder)

10 / 2 = 5 (tanpa sisa/remainder)

10 / 5 = 2 (tanpa sisa/remainder)

10 / 10 = 1 (tanpa sisa/remainder)

Sedangkan 3 bukan faktor dari 10, sebab 10 / 3 menyisakan 1.

Mathematical Term Penting untuk memahami RSA Algorithm

Common Factor / Faktor Persekutuan

Faktor Persekutuan dalam Matematika adalah angka **yang bisa membagi dua angka lain atau lebih** angka lain tanpa sisa/remainder.

Contoh 1:

Faktor dari 10 adalah 1, 2, 5, 10

Faktor dari 12 adalah 1, 2, 3, 4, 6, 12

Sehingga Common Factor/ Faktor Persekutuan dari 10 dan 12 adalah 1 dan 2, sebab hanya 1 dan 2 (faktor yang sama dari 10 dan 12) yang bisa membagi 10 dan 12 tanpa sisa.

Contoh 2:

Faktor dari 10 adalah 1, 2, 5, 10

Faktor dari 50 adalah 1, 2, 5, 10, 25, 50

Sehingga Common Factor/ Faktor Persekutuan dari 10 dan 50 adalah 1, 2, 5, dan 10.

Mathematical Term Penting untuk memahami RSA Algorithm

Prime Number / Bilangan Prima

Bilangan Prima adalah bilangan positif yang hanya bisa dibagi oleh angka 1 dan dirinya sendiri (memiliki faktor 1 dan dirinya sendiri). 1 bukan Bilangan Prima, sebab angka 1 hanya memiliki faktor dirinya sendiri.

Contoh:

2, 3, 5, 7, 11, 13, 29 …..

11 adalah Bilangan Prima, sebab 11 hanya habis dibagi 1 dan 11. Dalam artian lain, 11 hanya memiliki faktor 1 dan 11.

11 / 1 = 11

11 / 11 = 1

15 bukan Bilangan Prima, sebab 15 selain bisa dibagi 1 dan 15, dia juga bisa dibagi dengan 3, dan 5.

Dalam arti lain, 15 memiliki faktor selain 1 dan 15, yaitu 1, 3, 5, 15

Mathematical Term Penting untuk memahami RSA Algorithm

Semi Prime Number / Bilangan Semi Prima

Bilangan Semi Prima adalah bilangan positif yang hanya bisa dibagi oleh angka 1, dirinya sendiri dan Bilangan Prima.

Contoh:

21 adalah Bilangan Semi Prima, sebab angka 21 habis dibagi 1, 21 dan Bilangan Prima 3 dan 7. Dengan kata lain 21 memiliki faktor 1, 3, 7, dan 21.

21 / 1 = 21

21 / 3 = 7

21 / 7 = 3

21 / 21 = 1

6 adalah Bilangan Semi Prima, sebab angka 6 habis dibagi 1, 6 dan Bilangan Prima 2 dan 3. Dengan kata lain 6 memiliki faktor 1, 2, 3, dan 6.

6 / 1 = 6

6 / 3 = 2

6 / 2 = 3

6 / 6 = 1

Mathematical Term Penting untuk memahami RSA Algorithm

Modulo/ Operasi Modulus

Dalam Matematika, operasi Modulo/ Modulus adalah operasi yang menghasilkan sisa pembagian dari bilangan dengan bilangan lain. Dalam Bahasa Pemrograman **dilambangkan dengan simbol %**.

Contoh:

5 % 2 = 1

5 % 3 = 2

10 % 3 = 1

Langkang-langkah Men-generate RSA Public dan Private Key

Memilih **2 Bilangan Prima secara acak**. **Catatan:** pada *real-world application*, 2 Bilangan Prima yang dipilih harus **sangat besar**.

Untuk memudahkan, kita akan memilih Bilangan Prima 2 dan 5.

P = 2 dan Q = 5

Langkang-langkah Men-generate RSA Public dan Private Key

Mendapatkan Modulus N, dimana N adalah *product* dari **P dan Q**.

N = P x Q

N = 2 x 5

N = 10

Untuk informasi anda, **Modulus N** adalah Bilangan Semi Prima. Sebab **N** memiliki faktor **1**, **N** (dirinya sendiri), **P** dan **Q**. Dimana **P** dan **Q** adalah Bilangan Prima. **Baca kembali *section*: Semi Prime Number / Bilangan Semi Prima**.

Langkang-langkah Men-generate RSA Public dan Private Key

Mendapatkan **Nilai Totient φ(N)**. Fungsi ini digunakan untuk menghitung berapa banyak bilangan **Coprime** dalam *range* kurang dari atau sama dengan **N**. Bilangan **Coprime** adalah bilangan yang tidak memiliki **Common Factor/ Faktor persekutuan** dengan **N**.

N = 10, hasil perkalian dari **P** dan **Q** dari slide sebelumnya

Bilangan yang kurang dari atau sama dengan **10** adalah **1, 2, 3, 4, 5, 6, 7, 8, 9, 10**

Dari angka diatas, angka mana saja yang memiliki Faktor Persekutuan dengan 10 selain 1?

Angka tersebut adalah **2, 4, 5, 6, 8, 10**.

Angka yang tidak memiliki Faktor Persekutuan dengan 10 (disebut juga **Coprime**) = **1, 3, 7, 9**.

Sehingga Totient dari 10, **φ(10) = 4**

Menghitung **Coprime** dengan formula **φ(N)**

φ(N) = (P-1)(Q-1)

φ(N) = (2-1)(5-1)

φ(N) = 1 x 4

φ(N) = 4

Langkang-langkah Men-generate RSA Public dan Private Key

Mendapatkan Public Key Exponent

Untuk memilih Public Key, beberapa persyaratan dibawah ini harus dipenuhi.

- Harus Bilangan Prima
- Harus lebih dari satu dan kurang dari Totient **φ(N).**

  1 < Public Key < φ(N)

- Bukan faktor dari **N** dan Totient **φ(N).** Dalam arti lain, **Public Key** haruslah **Coprime** dengan **N** dan Totient **φ(N)**

  N % Public Key != 0 dan φ(N) % Public Key != 0

Dari persyaratan diatas, **katakanlah** kita akan memilih angka **3**. Sebab, angka **3** masuk dalam kategori tersebut.

Sehingga Public Key yang kita pilih adalah **3**.

Public Key Exponent = 3

Langkang-langkah Men-generate RSA Public dan Private Key

Mendapatkan Private Key Exponent

Untuk memilih Private Key, beberapa persyaratan dibawah ini harus dipenuhi.

- **(Public Key X Private Key) % φ(N) = 1**

  Dimana sejauh ini kita mendapatkan beberapa nilai:

  N = 10

  Totient φ(N) = 4

  Public Key Exponent = 3

  (3 x Private Key) % 4 = 1

Langkang-langkah Men-generate RSA Public dan Private Key

Mendapatkan Private Key Exponent

Untuk mensimulasikan **(3 x Private Key) % 4 = 1** kita akan membuat program sederhana

```
package main

import "fmt"

func main() {
        pk := 3
        t := 4
        for i := 1; i < 17; i++ {
                x := (pk * i) % t
                fmt.Println(x, " -> ", i)
        }
}
```

```
Output:
x =  3  Private Key ->  1
x =  2  Private Key ->  2
x =  1  Private Key ->  3
x =  0  Private Key ->  4
x =  3  Private Key ->  5
x =  2  Private Key ->  6
x =  1  Private Key ->  7
x =  0  Private Key ->  8
x =  3  Private Key ->  9
x =  2  Private Key ->  10
x =  1  Private Key ->  11
x =  0  Private Key ->  12
x =  3  Private Key ->  13
x =  2  Private Key ->  14
x =  1  Private Key ->  15
x =  0  Private Key ->  16
```

Pada **output** diatas kita mendapatkan beberapa angka yang memenuhi persyaratan **(3 x Private Key) % 4 = 1**. Perhatikan pada semua **output** ketika **x = 1**, kita mendapatkan **3, 7, 11, 15**. Keempat angka tersebut memenuhi syarat untuk menjadi **Private Key Exponent**. Kita akan memilih secara acak, katakanlah kita memilih **7** sebagai Private Key Exponent.

Private Key Exponent = 7

Langkang-langkah Men-generate RSA Public dan Private Key

Sejauh ini kita sudah mendapatkan semua komponen yang dibutuhkan.

P = 2

Q = 5

Modulus N = 10

Totient N, φ(10) = 4

Public Key Exponent = 3

Private Key Exponent = 7

Proses Encryption dengan RSA Public Key

Untuk melakukan Enkripsi, kita akan menggunakan Public Key yang sudah kita dapatkan.

Proses Enkripsi dinotasikan pada formula berikut

Encrypt(Message) = (Message ^ Public Key) % N

Catatan: **^** adalah simbol untuk pemangkatan. Sehingga **x ^ y** adalah x dipangkatkan dengan y

Contoh:

Message = 8

Encrypt(8) = (8 ^ 3) % 10

Encrypt(8) = 2

Hasil dari Enkripsi dengan Message = 8 menghasilkan Cipher Data = 2

Proses Decryption dengan RSA Private Key

Untuk melakukan Dekripsi, kita akan menggunakan Private Key yang sudah kita dapatkan.

Proses Dekripsi dinotasikan pada formula berikut

Decrypt(Cipher Data) = (Cipher Data ^ Private Key) % N

Contoh Cipher Data hasil enkripsi sebelumnya = 2:

Cipher Data = 2

Decrypt(2) = (2 ^ 7) % 10

Decrypt(2) = 8

Hasil dari Dekripsi dengan Cipher Data = 2 menghasilkan kembali Message = 8

RSA Key File dan Base64 Encoding

Jika anda pernah menggunakan algoritma RSA sebelumnya, pasti anda familiar dengan bentuk Private dan Public Key dari RSA.

```
-----BEGIN RSA PRIVATE KEY-----
MIICWwIBAAKBgQCbhSeJwoX0HClw/xGUH7cJ2GMk36o
QNA3ltLs9wsdEICrpc+Hb
```

```
-----BEGIN PUBLIC KEY-----
MIGfMA0GCSqGSIb3DQEBAQUAA4GNADCBiQKBgQCb
hSeJwoX0HClw/xGUH7cJ2GMk
```

Bentuk diatas adalah format Private dan Public Key RSA yang sudah *encode* ke dalam **Base64** format yang menyimpan komponen-komponen berikut.

Komponen Public Key

- Modulus N (Pada contoh materi ini N = 10)
- Public Key Exponent (Pada contoh materi ini Public Key Exponent = 3)

Komponen Private Key

- Modulus N (Pada contoh materi ini N = 10)
- Private Key Exponent (Pada contoh materi ini Private Key Exponent = 7)
- Prime Factor P dan Q (Pada contoh materi ini P = 2 dan Q = 5)
- Totient N, $\varphi(N)$ (Pada contoh materi ini $\varphi(N) = 4$)