Implementasi Tanda Tangan Digital untuk Pengamananan Dokumen Digital pada Pelaksanaan Smart Governance

Natasya Jatiwicha Azzahra (18219065)
Program Studi Sistem dan Teknologi Informasi
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung, Jalan Ganesha 10 Bandung
18219065@std.stei.itb.ac.id

***Abstract*—Perkembangan teknologi dalam beberapa tahun terakhir mengakibatkan adanya pengembangan konsep *Smart City,* salah satunya *Smart Governance*. Hal ini mengakibatkan perlu adanya administrasi yang dikelola secara terintegrasi. Untuk mengatasi masalah pemalsuan dokumen pemerintahan dibuatlah solusi dengan melakukan verifikasi menggunakan tanda tangan digital. Tanda tangan yang dibuat menggunakan algoritma RSA dan fungsi SHA-256.**

***Keywords—Tanda tangan digital, dokumen digital, smart governance,* RSA, SHA-256**

I. PENDAHULUAN

Perkembangan teknologi dalam beberapa tahun terakhir mengalami banyak kemajuan. Salah satunya adalah adanya revolusi industri 4.0 yang memunculkan adanya konsep "*Smart City*". Konsep ini bertujuan untuk mengembangkan tata kelola infrastruktur yang cerdas menggunakan teknologi informasi dan komunikasi sesuai dengan kebutuhan yang ada.

Smart City di Indonesia sendiri memiliki beberapa objek penting diantaranya adalah masyarakat, pemerintah, dan infrastruktur. Adanya kolaborasi dari objek-objek tersebut dapat mendukung percepatan *Smart City* yang akan dibangun. Salah satu bentuk dari kolaborasi tersebut adalah penerapan *Smart Governance*.

Smart Governance sendiri adalah bentuk inovasi teknologi dalam bidang tata kelola pemerintahan. Pengimplementasian Smart Governance ini sendiri diharapkan dapat membentuk tata kelola administrasi yang lebih efektif, efisien, komunikatif, dan terintegrasi. Target dari Smart Governance ini sendiri adalah *Smart Bureaucracy*, *Smart Public Service*, dan *Smart Public Policy* [1]. Namun dalam perjalanannya, masih banyak masalah yang muncul. Diantaranya adalah masih banyak ditemukannya pelayanan publik konvensional, kurangnya komunikasi yang disampaikan, keterlambatan dokumen, ketidakpastian informasi pemerintahan, hingga pemalsuan dokumen pemerintah [2].

Untuk mencegah masalah-masalah diatas terulang kembali, dibutuhkan suatu sistem pengamanan dokumen pemerintah secara digital. Pada sistem pengamanan ini digunakan metode tanda tangan digital yang dicantumkan sebagai bagian dari kop dokumen sehingga instansi yang bersangkutan dapat mengecek keaslian (verifikasi) dokumen digital yang dimilikinya. Dengan adanya sistem ini diharapkan akan mendukung percepatan *Smart City* khususnya dalam bidang pemerintahan.

II. DASAR TEORI

A. Tanda Tangan Digital

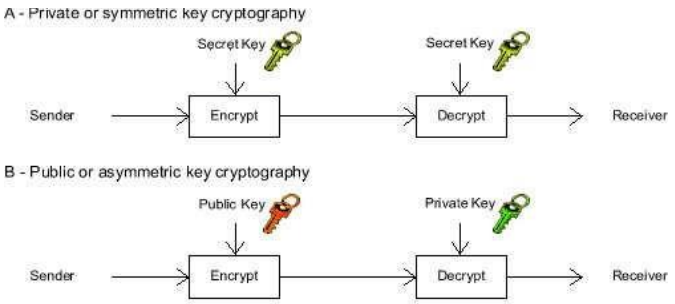
Tanda tangan digital merupakan sebuah skema matematis yang biasa digunakan sebagai memverifikasi keaslian suatu dokumen. Konsep tanda tangan digital ini cukup berbeda dengan tanda tangan yang dibubuhkan pada suatu dokumen. Tanda tangan digital adalah suatu nilai kriptografis yang bergantung ada isi pesan dan juga kuncinya [3]. Oleh karena itu, nilai dari tanda tangan digital pada suatu dokumen akan berubah jika terdapat perubahan pada teks dalam sebuah dokumen.

Konsep tanda tangan digital ini sesuai dengan konsep kriptografi yang menjamin beberapa aspek berikut: *confidentiality, authentication, data integrity, dan non-repudiation*. Keberadaan tanda tangan digital dalam suatu dokumen dapat menjadi salah satu mekanisme untuk menjamin pertanggungjawaban keabsahan dokumen.

Pada pembuatan tanda tangan digital terdapat beberapa karakteristik, diantaranya adalah sebagai berikut.

1. Tanda tangan adalah bukti yang otentik
2. Tanda tangan tidak dapat dilupakan
3. Tanda tangan tidak dapat dipindahkan untuk digunakan ulang
4. Dokumen yang telah ditandatangani dengan tidak dapat diubah
5. Tanda tangan tidak dapat disangkal [4]

Pada implementasinya sendiri, terdapat dua skema yang dapat digunakan. Diantaranya menggunakan kriptografi simetris dan juga asimetris. Alur kerja dari pengadaan tanda tangan digital dapat dilihat pada gambar II.1



Gambar II.1 Pengadaan tanda tangan digital. Sumber :https://www.researchgate.net/figure/Public-Private-Key-Encryption-and-Digital-Signatures_fig31_277736892

Pada makalah ini sendiri akan digunakan metode kriptografi asimetris (RSA) dan hash SHA-256 untuk menjaga autentikasi pada dokumen.

B. Kriptografi Asimetris (RSA)

Kunci asimetris (RSA) adalah pasangan kunci-kunci kriptografi yang salah satunya dipergunakan untuk proses enkripsi dan yang satu lagi untuk dekripsi [5]. Algoritma ini ditemukan oleh tiga peneliti MIT yaitu Ronald Rivest, Adi Shamir, dan Len Adleman. Keamanan sistem algoritma ini berada pada pemfaktoran bilangan prima yang bernilai sangat besar.

Semua orang yang mendapatkan kunci publik dapat menggunakannya untuk mengenkripsi suatu pesan, sedangkan hanya satu orang saja yang memiliki rahasia tertentu dalam hal ini kunci privat untuk melakukan pembongkaran terhadap sandi yang dikirim untuknya [3].

Algoritma RSA memiliki beberapa komponen, diantaranya:

- p dan q → bilangan prima & rahasia
- n = p * q → tidak rahasia
- ϕ(n) = (p-1)*(q-1) → rahasia
- e (kunci enkripsi) → tidak rahasia
- d (kunci dekripsi) → rahasia
- m (plaintext) → rahasia
- c (cyphertext) → tidak rahasia

Konsep yang digunakan pada tanda tangan digital cukup berbeda dengan yang dijelaskan sebelumnya. Penggunaannya dapat dibilang terbalik. Dimana kunci privat digunakan untuk melakukan enkripsi dan kunci publik dilakukan untuk melakukan dekripsi.

C. Hash SHA-256

Fungsi Hash sering disebut dengan fungsi satu arah *(one-way function*). *Message digest, fingerprint,* fungsi kompresi dan message authentication code (MAC), merupakan suatu fungsi matematika yang mengambil masukan panjang variabel dan mengubahnya ke dalam urutan biner dengan panjang yang tetap. Fungsi Hash biasanya diperlukan bila ingin membuat sidik jari dari suatu pesan. Sidik jari pada pesan merupakan suatu tanda bahwa pesan tersebut benarbenar berasal dari orang yang diinginkan [6].

Pada makalah ini digunakan algoritma SHA-256 untuk melakukan hashing terhadap dokumen. SHA-256 digunakan karena dianggap memiliki kemungkinan kolisi kecil. Kolisi sendiri adalah keadaan dimana nilai hash pada dua pesan berbeda memiliki hasil yang sama.

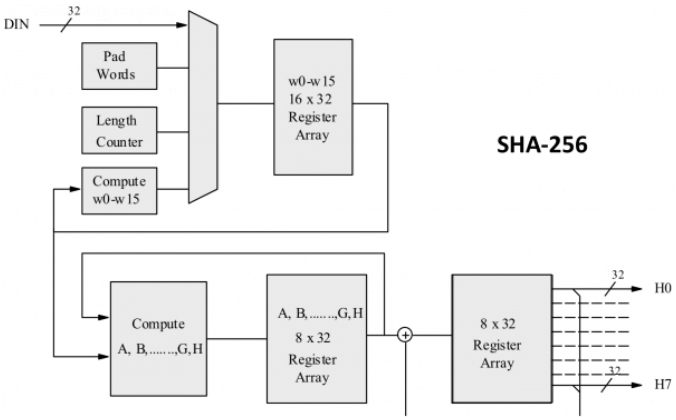
III. Rancangan Solusi dan Implementasi

Dalam menyelesaikan masalah-masalah dalam pelaksanaan *Smart Governance* digunakan sistem tanda tangan digital dengan fungsi enkripsi RSA dan hash SHA-256. Hal ini dilakukan untuk tetap menjaga prinsip-prinsip yang ada pada kriptografi.

Perancangan solusi dan implementasi ini terdiri dari beberapa tahapan, sebagai berikut.

A. Hashing

Pada tahap pertama akan dilakukan hashing terhadap dokumen yang ada menggunakan algoritma SHA-256. Langkah-langkah yang dilakukan pada hashing ini sendiri adalah sebagai berikut.



Gambar III.2 Algoritma SHA-256. Sumber: https://www.cast-inc.com/security/encryption-primitives/sha-256

1. L merupakan panjang pesan
2. Inisialisasi hash values (h) yang berisi 32-bit pertama dari pecahan akar dari 8 prima pertama (2…19)
3. Inisialisasi array berisi konstanta bilangan bulat yang berisi 32-bit pertama dari pecahan akar kubik dari 64 prima pertama (2…311)
4. Melakukan padding dengan menambahkan satu ‘1’ bit.
5. Melakukan padding dengan menambahkan ‘0’ bit sebanyak K dimana K merupakan angka minimum >= 0 yang memenuhi L + 1 + K + 64 % 512 = 0
6. Melakukan padding dengan menambahkan L sebagai 64-bit big-endian dimana total pesan yang telah ditambahkan padding menjadi kelipatan 512.
7. Membagi pesan yang telah di-padding menjadi blok berukuran 512

8. Untuk setiap blok, melakukan right rotate dan right shift untuk setiap karakter. Selanjutnya melakukan compression secara looping
9. Nilai yang sudah terkompresi kemudian akan dimodifikasi menjadi 32-bit untuk setiap nilai h dari h0 sampai h7 untuk menghasilkan hash value yang final. [7]

Implementasi pada bagian ini terlihat pada gambar dibawah ini.

```
Your hashed message is:  3b928b7b9f7c4e137dc7db0b80af7a044a51e83c5cf9
9a67a5167d2b85ee58c9
```

Gambar III.1 Implementasi Hashing

.

B. Pembangkitan Kunci Asimetri

Tahapan pembangkitan kunci ini dilakukan dengan inisialiasi nilai p, q, dan e. Nilai p dan q merupakan nilai bilangan prima. Sedangkan e berasal dari nilai yang relatif prima terhadap nilai (p-1)*(q-1).

Pembangkitan kunci diawali dengan mengubah pesan menjadi nilai integer (*array of integer*). Kemudian memecah *array of integer* tersebut menjadi beberapa blok yang sama panjangnya. Selanjutnya akan dilakukan enkripsi pada hasil hash yang telah dilakukan sebelumnya dengan persamaan sebagai berikut.

$$e^{-1} \bmod (\phi(n))$$

Dari persamaan diatas, hasil yang ada digabungkan menjadi sebuah *ciphertext*. Implementasi pada bagian ini terlihat pada gambar dibawah ini.

```
Enter a prime number (17, 19, 23, etc): 17
Enter another prime number (Not one you entered above): 23
Generating your public/private keypairs now . . .
Your public key is  (227, 391)  and your private key is  (107, 391)
```

Gambar III.2 Implementasi Pembangkitan Kunci

C. Penandatanganan Dokumen

Pada tahapan ini digunakan nilai hash yang telah dienkripsi. Ciphertext ini kemudian disisipkan pada file dokumen. Tanda tangan dipisahkan dengan nilai * pada awal dan akhir pesan. Implementasi pada bagian ini terlihat pada gambar dibawah ini.

```
Encrypting message with private key  (107, 391)  . . .
Number representation before encryption:  [51, 98, 57, 50, 56, 98, 55
, 98, 57, 102, 55, 99, 52, 101, 49, 51, 55, 100, 99, 55, 100, 98, 48,
 98, 56, 48, 97, 102, 55, 97, 48, 52, 52, 97, 53, 49, 101, 56, 51, 99
, 53, 99, 102, 57, 57, 97, 54, 55, 97, 53, 49, 54, 55, 100, 50, 98, 5
6, 53, 101, 101, 53, 56, 99, 57]
Your encrypted hashed message is:
30622510710111322513225107136133331822021330613349333133492259522511395329136133299518183291952132201133063331953331361071073292111332919521321113349101225113195220220195113333107
```

Gambar III.3 Implementasi Penandatangan Dokumen

Hasil akhir penandatanganan dokumen:

Dinas Pendidikan dan Kebudayaan
Kota Besar
Jl. Bersama No. 1, Kota Besar

Kota Besar, 17 Desember 2021
Nomor: 12.188/DP-KK/XII/2021

Lampiran: –

Perihal: Musyawarah Guru Mata Pelajaran
Yth. Guru Mata Pelajaran Matematika

Kelas XII seluruh sekolah di Kota Besar
Dengan hormat,
Bersama surat ini kami mengundang bapak/ibu guru mata pelajaran matematika kelas XII di seluruh tingkat sekolah menengah atas di Kota Besar untuk menghadiri Musyawarah Guru Mata Pelajaran yang akan dilaksanakan pada:

Hari/Tanggal: Rabu, 22 Desember 2021
Pukul: 13.00–selesai
Tempat: Aula SMA Maju Bersama Kota Besar
Acara:
1. Soal Penilaian Akhir Semester mata pelajaran matematika
2. Pembahasan praktik KBM dengan pertemuan tatap muka terbatas
3. Sosialisasi kurikulum paradigma baru
Demikian undangan ini kami sampaikan agar dapat dihadiri oleh perwakilan guru mata pelajaran matematika dari setiap sekolah.

Terima kasih atas perhatian yang diberikan.

Tembusan:
Kepala Sekolah Menengah Atas se-Kota Besar

Ketua MGMP,

Sajak Bersahaja

30622510710111322513225107136133331822021330613349333133492259522511395329136133299518183291952132201133063331953331361071073292111332919521321113349101225113195220220195113333107

D. Validasi Dokumen

Tahapan ini dilakukan dengan mengambil kembali nilai tanda tangan digital yang sudah dibuat. Tanda tangan ini akan dilakukan dekripsi dengan menggunakan kunci publik instansi pembuat. Hasil dari nilai dekripsi akan dibandingkan dengan nilai *hash* dokumen yang dikirimkan. Implementasi pada bagian ini terlihat pada gambar dibawah ini.

```
Decrypting message with public key  (227, 391)  . . .
Decrypted number representation is:  [51, 98, 57, 50, 56, 98, 55, 98,
 57, 102, 55, 99, 52, 101, 49, 51, 55, 100, 99, 55, 100, 98, 48, 98,
56, 48, 97, 102, 55, 97, 48, 52, 52, 97, 53, 49, 101, 56, 51, 99, 53,
 99, 102, 57, 57, 97, 54, 55, 97, 53, 49, 54, 55, 100, 50, 98, 56, 53
, 101, 101, 53, 56, 99, 57]
Your decrypted message is:
3b928b7b9f7c4e137dc7db0b80af7a044a51e83c5cf99a67a5167d2b85ee58c9

Verification process . . .
Verification successful:
a67a5167d2b85ee58c9
a67a5167d2b85ee58c9
```

Gambar III.4 Implementasi Validasi

IV. KESIMPULAN

Berdasarkan hasil pengerjaan yang telah dilakukan, didapatkan kesimpulan bahwa penggunaan tanda tangan digital dapat dilakukan dalam pengembangan *Smart City*, khususnya *Smart Governance*. Pengembangan pada makalah ini terbatas pada pengecekan dokumen antar institusi. Pengembangan yang disebarluaskan pada masyarakat masih diperlukan pengembangan kedepannya dikarenakan masalah keamanan.

Kedepannya solusi ini diharapkan dapat dikembangkan lebih lagi, terkhusus pada bagian pengamanan dokumen sehingga dokumen-dokumen negara tidak disalahgunakan oleh pihak yang tidak berkepentingan.

VIDEO LINK AT YOUTUBE

Link Video: https://bit.ly/tugasmakalahnatasya

UCAPAN TERIMA KASIH

Ucapan terima kasih penulis nyatakan kepada Tuhan Yang Maha Esa, karena karunia-Nya penulis bisa diberikan kesempatan untuk menyelesaikan dan bisa memberikan kontribusi nyata dalam memberikan ide yang dituliskan pada makalah ini. Penulis juga mengucapkan terima kasih kepada Dr. Rinaldi Munir atas dedikasinya dalam memberikan ilmu pengetahuan terkait mata kuliah kriptografi kepada penulis.

REFERENCES

[1] Smartcity Kulon Progo. (n.d.). *Smart Governance*. 6 Dimensi Kulonprogo Smartcity. Retrieved May 25, 2022, from http://smartcity.kulonprogokab.go.id/smart_governance

[2] Smartcity Kulon Progo. (n.d.). *Smart Governance*. 6 Dimensi Kulonprogo Smartcity. Retrieved May 25, 2022, from http://smartcity.kulonprogokab.go.id/smart_governance

[3] *Ombudsman RI: Smart City dan Smart Governance Harus Berjalan Bersamaan*. (2021, July 8). Ombudsman RI. Retrieved May 8, 2022, from https://ombudsman.go.id/news/r/ombudsman-ri-smart-city-dan-smart-governance-harus-berjalan-bersamaan

[4] Munir, Rinaldi. 2020. Slide Kuliah IF4020 Kriptografi: Tanda Tangan Digital

[5] Munir, Rinaldi. 2020. Slide Kuliah IF4020 Kriptografi: Tanda Tangan Digital

[6] Basri, "Kriptografi Simetris Dan Asimetris Dalam Perspektif Keamanan Data Dan Kompleksitas Komputasi," J. Ilm. Ilmu Komput., vol. 2, no. 2, pp. 2442–4512, 2016.

[7] *256-bit SHA Secure Hash Crypto Engine*. (n.d.). Cast Silicon IP Cores. Retrieved June 25, 2022, from https://www.cast-inc.com/security/encryption-primitives/sha-256

[8] D. Ariyus, "Pengantar Ilmu Kriptografi Teori, Analisis dan Implementasi.," Journal of Chemical Information and Modeling. pp. 1689–1699, 2008.

PERNYATAAN

Dengan ini saya menyatakan bahwa makalah yang saya tulis ini adalah tulisan saya sendiri, bukan saduran, atau terjemahan dari makalah orang lain, dan bukan plagiasi.

Bandung, 26 April 2021

Natasya Jatiwicha,18219065

Appendix

```
import random
from hashlib import sha256


def coprime(a, b):
    while b != 0:
        a, b = b, a % b
    return a


def extended_gcd(aa, bb):
    lastremainder, remainder = abs(aa),
abs(bb)
    x, lastx, y, lasty = 0, 1, 1, 0
    while remainder:
        lastremainder, (quotient,
remainder) = remainder,
divmod(lastremainder, remainder)
        x, lastx = lastx - quotient*x, x
        y, lasty = lasty - quotient*y, y
    return lastremainder, lastx * (-1 if aa
< 0 else 1), lasty * (-1 if bb < 0 else 1)

#Euclid's extended algorithm for finding
the multiplicative inverse of two
numbers
def modinv(a, m):
    g, x, y = extended_gcd(a, m)
    if g != 1:
        raise Exception('Modular inverse
does not exist')
    return x % m


def is_prime(num):
    if num == 2:
        return True
    if num < 2 or num % 2 == 0:
        return False
    for n in range(3, int(num**0.5)+2, 2):
        if num % n == 0:
            return False
    return True
```

```
def generate_keypair(p, q):
    if not (is_prime(p) and is_prime(q)):
        raise ValueError('Both numbers must
be prime.')
    elif p == q:
        raise ValueError('p and q cannot be
equal')

    n = p * q

    #Phi is the totient of n
    phi = (p-1) * (q-1)

    #Choose an integer e such that e and
phi(n) are coprime
    e = random.randrange(1, phi)

    #Use Euclid's Algorithm to verify that
e and phi(n) are comprime
    g = coprime(e, phi)

    while g != 1:
        e = random.randrange(1, phi)
        g = coprime(e, phi)

    #Use Extended Euclid's Algorithm to
generate the private key
    d = modinv(e, phi)

    #Return public and private keypair
    #Public key is (e, n) and private key
is (d, n)
    return ((e, n), (d, n))


def encrypt(privatek, plaintext):
    #Unpack the key into it's components
    key, n = privatek

    #Convert each letter in the plaintext
to numbers based on the character using a^b
mod m

    numberRepr = [ord(char) for char in
plaintext]
```

```
    print("Number representation before
encryption: ", numberRepr)
    cipher = [pow(ord(char),key,n) for char
in plaintext]

    #Return the array of bytes
    return cipher


def decrypt(publick, ciphertext):
    #Unpack the key into its components
    key, n = publick

    #Generate the plaintext based on the
ciphertext and key using a^b mod m
    numberRepr = [pow(char, key, n) for
char in ciphertext]
    plain = [chr(pow(char, key, n)) for
char in ciphertext]

    print("Decrypted number representation
is: ", numberRepr)

    #Return the array of bytes as a string
    return ''.join(plain)


def hashFunction(message):
    hashed = sha256(message.encode("UTF-
8")).hexdigest()
    return hashed


def verify(receivedHashed, message):
    ourHashed = hashFunction(message)
    if receivedHashed == ourHashed:
        print("Verification successful: ",
)
        print(receivedHashed, " = ",
ourHashed)
    else:

        print("Verification failed")
        print(receivedHashed, " != ",
ourHashed)
```

```
def main():
    p = int(input("Enter a prime number
(17, 19, 23, etc): "))
    q = int(input("Enter another prime
number (Not one you entered above): "))
    #p = 17
    #q=23


    print("Generating your public/private
keypairs now . . .")
    public, private = generate_keypair(p,
q)

    print("Your public key is ", public ,"
and your private key is ", private)
    message = input("Enter a message to
encrypt with your private key: ")
    print("")

    hashed = hashFunction(message)
    print("Your hashed message is: ",
hashed)

    print("Encrypting message with private
key ", private ," . . .")
    encrypted_msg = encrypt(private,
hashed)
    print("Your encrypted hashed message
is: ")
    print(''.join(map(lambda x: str(x),
encrypted_msg)))
    #print(encrypted_msg)

    print("")
    print("Decrypting message with public
key ", public ," . . .")

    decrypted_msg = decrypt(public,
encrypted_msg)
    print("Your decrypted message is:")
    print(decrypted_msg)

    print("")
    print("Verification process . . .")
```

```
    verify(decrypted_msg, message)

main()
```