Implementasi Skema Schnorr pada Tanda Tangan Digital

Chintya Wijaya (18219082)
Program Studi Sistem dan Teknologi Informasi
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung, Jalan Ganesha 10 Bandung
E-mail: cwchintya@gmail.com

***Abstract*—Pandemi COVID-19 memiliki dampak pada maraknya transformasi digital. Kegiatan yang sebelumnya hanya dilakukan secara luring berubah menjadi daring sejak pandemi terjadi. Salah satu aktivitas terdampak adalah penandatanganan dokumen. Tanda tangan umumnya dilakukan secara langsung di atas kertas. Interaksi yang terbatas selama pandemi membuat penandatanganan dokumen secara digital menjadi lebih banyak dilakukan. Tanda tangan digital yang telah diakui secara hukum dapat menjamin keaslian informasi. Salah satu skema tanda tangan digital adalah skema tanda tangan Schnorr. Skema tanda tangan digital ini diklaim memiliki performa yang lebih cepat dibandingkan skema lainnya. Oleh karena itu, penulis melakukan eksperimen pada skema tersebut dibandingkan dengan tanda tangan digital dengan *hash* SHA-1 dan algoritma RSA. Dari implementasi tersebut, dapat dibandingkan performa skema tanda tangan digital Schnorr dibanding skema tanda tangan digital lain sehingga dapat ditarik kesimpulan penggunaan skema tanda tangan digital yang umum digunakan dibanding skema Schnorr.**

Keywords—tanda tangan digital, skema tanda tangan Schnorr, hash, kriptografi

I. Pendahuluan

Pandemi COVID-19 yang sudah berjalan lebih dari dua tahun memiliki dampak pada kegiatan masyarakat. Kegiatan yang sebelumnya hanya dilakukan secara luring kini dilakukan secara daring karena terbatasnya interaksi langsung saat pandemi. Transformasi digital semakin digencarkan mengingat digitalisasi merupakan solusi dari terbatasnya interaksi akibat pandemi.

Salah satu kegiatan yang terdampak adalah penandatanganan dokumen. Tanda tangan digunakan untuk menjamin keaslian dokumen, baik keaslian informasi di dalamnya maupun orang yang menandatangani dokumen [1]. Tanda tangan umumnya dilakukan secara langsung di atas kertas karena memiliki kekuatan hukum. Akibat terbatasnya interaksi langsung saat pandemi, tanda tangan digital menjadi solusi pemenuhan legalitas dokumen.

Skema tanda tangan digital Schnorr merupakan salah satu skema tanda tangan digital yang menggunakan fungsi hash. Skema ini dapat menghasilkan ukuran tanda tangan yang lebih kecil [2]. Oleh karena itu, pada makalah ini akan dibahas mengenai implementasi skema tanda tangan digital Schnorr meliputi teori dan pembuatan program tanda tangan digital.

II. Metode

A. *Literature Review*

Metode pertama adalah *literature review* dengan meninjau berbagai sumber baik buku, materi ajar, publikasi, dan *website* terkait tanda tangan digital dengan skema Schnorr. Kemudian, akan ditemukan dan diambil informasi terkait skema tanda tangan digital Schnorr untuk lanjut ke metode berikutnya.

B. Eksperimen

Metode kedua adalah eksperimen, yaitu membuat implementasi program skema tanda tangan digital Schnorr dengan bahasa pemrograman Python. Dari program yang dibuat akan dilihat performa dari skema Schnorr dibandingkan dengan skema tanda tangan digital lain dengan RSA dan SHA-1 pada tugas kecil sebelumnya. Seluruh program akan ditulis dalam bahasa pemrograman Python menggunakan *library* yang sudah tersedia secara publik.

III. Dasar Teori

Pada bagian ini akan dibahas mengenai konsep yang melatarbelakangi makalah ini.

A. Kriptografi

Kriptografi merupakan ilmu dan seni untuk menjaga keamanan pesan. Aspek keamanan yang ditangani oleh kriptografi diantaranya *confidentiality*, *integrity*, *authentication*, dan *non-repudiation*. *Confidentiality* berarti pesan terjaga kerahasiannya. *Integrity* berarti pesan terjaga keasliannya. *Authentication* berarti pengirim dan penerima pesan merupakan pihak yang sah dan tidak menyamar. *Non-repudiation* berarti tidak ada penyangkalan dari pengirim pesan saat pesan sudah dikirimkan [1].

Pesan dalam kriptografi dapat dikategorikan menjadi dua, yaitu *plaintext* dan *ciphertext*. *Plaintext* merupakan pesan yang memiliki makna, sedangkan *ciphertext* tidak memiliki makna. Enkripsi merupakan proses menyandikan *plaintext* menjadi *ciphertext*, sedangkan dekripsi merupakan proses

mengembalikan *ciphertext* menjadi *plaintext* dengan bantuan kunci.

Sistem kriptografi tergolong menjadi dua, yaitu kriptografi kunci simetri dan kriptografi kunci publik. Kriptografi kunci simetri menggunakan kunci rahasia yang sama untuk proses enkripsi dan dekripsi, sedangkan pada kriptografi kunci publik digunakan kunci yang berbeda. Kunci berbeda tersebut terdiri dari kunci publik untuk proses enkripsi (tidak rahasia) dan kunci privat untuk proses dekripsi (rahasia).

B. Fungsi *Hash*

Fungsi *hash* merupakan fungsi yang mengonversi pesan menjadi *string* berukuran tetap, yang biasa disebut sebagai *message-digest* (pesan ringkas) atau *hash value* (nilai hash) [1]. Fungsi *hash* dinilai aman karena nilai *hash* tidak dapat dikembalikan ke pesan semula. Hal ini terkait dengan sifat fungsi *hash* seperti:

1. *Collision resistance*
   Collision resistance berarti sangat sulit menemukan *input* suatu *hash* yang akan menghasilkan *message digest* yang sama.
2. *Preimage resistance*
   Preimage resistance berarti sulit menemukan *input* suatu *hash* apabila diketahui *message digest*-nya.
3. *Second preimage resistance*
   Second preimage resistance berarti untuk *input* suatu *hash* dan *message digest*-nya, sulit menemukan *input* lainnya yang akan menghasilkan *message digest* yang sama.

Terdapat beberapa jenis fungsi *hash* dalam kriptografi, seperti MD2/MD4/MD5, SHA-1, SHA-256, SHA-512, SHA-3, dan lain-lain. Selanjutnya akan dibahas mengenai algoritma salah satu *hash*, yaitu SHA-1.

SHA-1 atau *Secure Hash Algorithm* merupakan sebuah fungsi *hash* kriptografis yang menghasilkan 160-bit *message digest*. Berikut adalah *pseudocode* dari SHA-1 [3].

***Pseudocode* SHA-1**

Misalkan sebuah pesan *string* "abc" akan dimasukkan dalam fungsi *hash* SHA-1, dalam bilangan biner menjadi

01100001 01100010 01100011

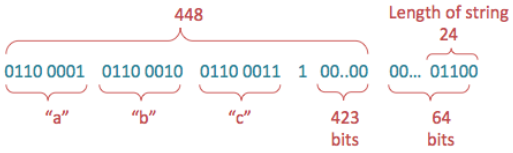
dan dalam heksadesimal menjadi

616263.

1. Inisialisasi 5 *random string* dalam heksadesimal, misalnya

   H_0 = 67DE2A01
   H_1 = BB03E28C
   H_2 = 011EF1DC
   H_3 = 9293E9E2
   H_4 = CDEF23A9

2. Pesan kemudian diberi *padding* dengan menyisipkan '1' diikuti dengan '0' hingga mencapai 448-bit pesan. 64-bit berikutnya ditambahkan dengan panjang *string* pesan. Contoh *string* "abc" memiliki 24 bit, seperti ilustrasi ini.



3. *Input* dengan *padding* tersebut disebut sebagai M, dibagi menjadi 512-bit *chunks*, dan setiap *chunk*-nya dibagi menjadi enam belas 32-bit *words*, W_0, …, W_{15}. Pada *string* "abc" hanya terdapat satu *chunk* karena memiliki kurang dari 512-bit pesan.

4. Iterasi sebanyak 80 kali akan dimulai untuk setiap *chunk* i, M_n:
   a. Iterasi *chunk* 16 hingga 79 (inklusif) berlaku

      $W(i) = S^1 (W(i-3)$ xor $W(i-8)$ xor $W(i-14)$ xor $W(i-16))$

   b. *Circular shift* $S^n (X)$ pada X dalam n bit, dengan n bilangan *integer* antara 0 dan 32

      $$S^n(X) = (X << n) \text{ or } (X >> 32 – n)$$

      X << n adalah operasi *shift-left* dan X >> 32 – n adalah operasi *shift-right*.

5. Simpan nilai *hash* dalam variable berikut

   $A = H_0$
   $B = H_1$
   $C = H_2$
   $D = H_3$
   $E = H_4$

6. Untuk 80 iterasi dengan i diantara 0 dan 79 (inklusif), hitung

   $$TEMP = S^5 * (A) + f(i; B, C, D) + E + W(i) + K(i)$$

   Hitung kembali:

   $E = D$
   $D = C$
   $C = S^{30} (B)$
   $B = A$
   $A = TEMP$

   dengan fungsi f berlaku:

$$f(i;B,C,D) = (B \wedge C) \vee ((\neg B) \wedge D) \quad \text{for } 0 \geq i \geq 19$$

$$f(i;B,C,D) = B \oplus C \oplus D \quad \text{for } 20 \geq i \geq 39$$

$$f(i;B,C,D) = (B \wedge C) \vee (B \wedge D) \vee (C \wedge D) \quad \text{for } 40 \geq i \geq 59$$

$$f(i;B,C,D) = B \oplus C \oplus D \quad \text{for } 60 \geq i \geq 79.$$

7. Simpan hasil *hash* pada *chunk* dan lanjutkan pada *chunk* berikutnya:

$$H_0 = H_0 + A$$
$$H_1 = H_1 + B$$
$$H_2 = H_2 + C$$
$$H_3 = H_3 + D$$
$$H_4 = H_4 + E$$

8. Setelah dilakukan pada seluruh *chunk*, *message digest* merupakan *string* berukuran 160 bit berikut:

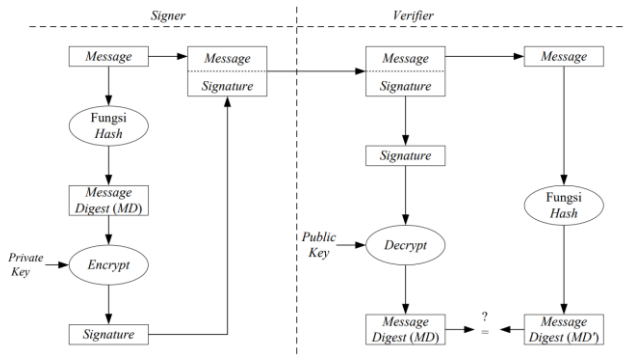
$$HH = S^{128}(H_0) \text{ or } S^{96}(H_1) \text{ or } S^{64}(H_2) \text{ or } S^{32}(H_3) \text{ or } H_4$$

String "abc" setelah di-*hash* akan menjadi a9993e364706816aba3e25717850c26c9cd0d89d

Gambar 1 Algoritma SHA-1
(sumber: https://brilliant.org/wiki/secure-hashing-algorithms/#:~:text=SHA%2D1%20works%20by%20feeding, encrypt%20messages%20using%20SHA%2D1.)

C. Tanda Tangan Digital

Tanda tangan digital merupakan tanda tangan untuk data digital yang digunakan untuk menjamin otentikasi, keaslian, dan anti-penyangkalan pesan dalam kriptografi. Umumnya, tanda tangan digital menggunakan fungsi *hash* dan algoritma kunci publik. Alur tanda tangan digital dapat dilihat pada gambar berikut.



Gambar 2 Diagram alir tanda tangan digital
(sumber: https://informatika.stei.itb.ac.id/~rinaldi.munir/Kriptografi-dan-Koding/2021-2022/21%20-%20Tanda-tangan-digital-2021.pdf)

Terdapat *signer* dan *verifier* dalam diagram alir di atas. *Signer* akan menerapkan fungsi *hash* pada pesan yang ingin ditandatangani dan mendapatkan *message digest*. Hasil tersebut akan dienkripsi dengan kunci privat milik *signer* menjadi sebuah tanda tangan digital yang disisipkan dalam pesan.

Untuk melakukan verifikasi, *verifier* akan mendapatkan pesan dan dimasukkan pada fungsi *hash* dan menghasilkan *message digest*. Hasil *message digest* kemudian dibandingkan dengan tanda tangan yang didekripsi dengan kunci publik pengirim. Apabila perbandingan sama, pengirim terautentikasi dan pesan yang diterima merupakan pesan asli.

Tanda tangan digital kini telah memiliki kekuatan hukum, tertulis dalam Peraturan Pemerintah tentang Penyelenggaraan Sistem dan Transaksi Elektronik.

IV. Pembahasan

A. Skema Tanda Tangan Digital Schnorr

Salah satu skema tanda tangan digital adalah skema tanda tangan digital Schnorr. Skema tanda tangan Schnorr dikenalkan pada tahun 1989 oleh Claus Schnorr. Skema tanda tangan ini diadaptasi dari skema tanda tangan ElGamal dan menghasilkan ukuran tanda tangan yang lebih kecil.

Algoritma skema tanda tangan digital Schnorr umumnya memiliki langkah yang sama dengan algoritma tanda tangan digital, yaitu:

1. Pengirim akan membangkitkan kunci

   Proses pembangkitan kunci oleh pengirim akan menghasilka kunci publik dan kunci privat. Algoritma pembangkitan kunci mengadaptasi algoritma skema tanda tangan ElGamal, dapat dilihat pada gambar berikut.

Algoritma Pembangkitan Kunci

Input : bilangan prima p dan q, bilangan a, elemen primitif α_0.

Output : kunci publik (p, q, α, β) dan kunci privat a.

Langkah:

1. Pilih bilangan prima p dan q sehingga $p - 1 \equiv 0 \pmod q$
2. Tentukan elemen primitif α_0 dari $\mathbb{Z}_p^*$ (sekumpulan bilangan yang kongruen dan relatif prima dengan p) dan hitung $\alpha = \alpha_0^{(p-1)/q} \pmod p$
3. Pilih sebarang bilangan bulat a dengan $0 \leq a \leq q - 1$
4. Hitung $\beta = \alpha^a \pmod p$
5. Diperoleh kunci publik (p, q, α, β) dan kunci privat a (rahasia)

Gambar 3 Algoritma pembangkitan kunci skema tanda tangan digital Schnorr

2. Tanda tangan digital dengan fungsi *hash*

Tanda tangan digital dengan skema Schnorr menggunakan fungsi *hash* untuk mengurangi ukuran pesan. Sebelum proses penandatanganan, dilakukan perhitungan nilai α^k mod p, dengan α dan p kunci publik dan k bilangan rahasia $1 \leq k \leq q - 1$. Hasil perhitungan akan digabungkan dengan pesan yang akan ditandatangani untuk dimasukkan pada fungsi *hash*. Fungsi *hash* yang digunakan dapat dilihat pada gambar berikut.

Algoritma Perhitungan *Hash*
Input : pesan, nilai α^k mod p
Output : nilai *hash*

Langkah:
1. Memotong pesan D menjadi blok $d_1, d_2, d_3, \ldots, d_n$ dalam ukuran yang sama sehingga satu bloknya merupakan satu karakter
2. Menggabungkan nilai α^k mod p pada blok-blok pesan menjadi $d_1, d_2, d_3, \ldots, d_m, s_1, s_2, s_3, \ldots, s_n$
3. Konversi potongan pesan ke ASCII
4. Menghitung nilai hash h (D | α^k mod p) = $(d_1 + d_2 + d_3 + \ldots + d_m + s_1 + s_2 + s_3 + \ldots + s_n) \bmod 277 + 1$
5. Didapatkan nilai hash

Gambar 4 Algoritma pembangkitan kunci skema tanda tangan digital Schnorr

Kemudian akan didapatkan nilai *hash* untuk digunakan dalam proses tanda tangan yang dapat dilihat pada gambar berikut.

Algoritma Tanda Tangan Digital
Input : kunci publik (p, q, α, β), kunci privat a, dan bilangan acak k
Output : tanda tangan untuk pesan (γ, δ)

Langkah:
1. Pilih bilangan acak k dengan $1 \leq k \leq q - 1$, dan k merupakan bilangan rahasia
2. Gunakan kunci publik α dan p, hitung nilai α^k mod p, kemudian digabung dengan pesan
3. Hitung nilai hash h (α^k mod p, D) dengan D adalah pesan, yang akan menjadi tanda tangan pertama (γ)
4. Hitung tanda tangan kedua $\delta = k + a\,\gamma \pmod q$
5. Diperoleh tanda tangan (γ, δ)

Gambar 5 Algoritma tanda tangan skema tanda tangan digital Schnorr

3. Verifikasi tanda tangan oleh penerima

Verifikasi dilakukan oleh penerima pesan untuk memastikan keaslian pesan dan autentikasi pengirim. Algoritma verifikasi pesan dapat dilihat pada gambar berikut.

Algoritma Verifikasi Tanda Tangan
Input : kunci publik α, β, p, kunci privat a, tanda tangan (γ, δ), pesan D, dan bilangan acak rahasia k
Output : hasil perhitungan $\alpha^\delta \beta^{-\gamma}$ mod p = a^k mod p, h ($\alpha^\delta \beta^{-\gamma}$ mod p, M) = γ

Langkah:
1. Hitung $\alpha^\delta \beta^{-\gamma}$ mod p
2. Pastikan $\alpha^\delta \beta^{-\gamma}$ mod p = a^k mod p
3. Hitung nilai hash h ($\alpha^\delta \beta^{-\gamma}$ mod p, D)
4. Pastikan nilai hash h($\alpha^\delta \beta^{-\gamma}$ mod p, D) = γ
5. Jika kedua perbandingan sama, tanda tangan asli dan pesan belum berubah
6. Jika salah satu perbandingan tidak sama, tanda tangan palsu dan pesan telah berubah

Gambar 6 Algoritma verifikasi tanda tangan skema tanda tangan digital Schnorr

B. Desain Program

Implementasi skema tanda tangan digital Schnorr dilakukan dengan bahasa pemrograman Python. Alasan pemilihan bahasa didasarkan pada kelengkapan *library* dan dokumentasi terkait implementasi.

Library dan sumber yang digunakan dalam implementasi program tanda tangan digital Schnorr terdiri dari Program RSA tugas kecil mata kuliah II4031 Kriptografi dan Koding STI milik kelompok penulis.

Spesifikasi program tanda tangan yang dibuat meliputi:

1. Dokumen dianggap sebagai data tipe String, sehingga tidak berupa dokumen sebenarnya
2. Nilai p dan q dianggap tetap
3. Program dieksekusi dari terminal, tidak menggunakan GUI
4. Program dapat menghitung waktu eksekusi sebagai perbandingan skema tanda tangan digital Schnorr dan skema sebelumnya

Output program dari makalah ini terdiri dari program dengan skema tanda tangan digital Schnorr dan program tanda tangan digital sebelumnya (tugas kecil). Kode program berdasarkan algoritma yang dibahas sebelumnya dapat dilihat pada bagian berikut.

1) Source Code Pembangkitan Kunci

Berikut merupakan *source code* pembangkitan kunci.

Kode Program Pembangkitan Kunci

```
def generatekey (p,q,a):
    alpha0 = 2
    alpha = (alpha0**((p-1)/q)%p
    beta = (alpha**a)%p

    pubkey = (p,q,alpha,beta)
    privkey = a

    return (pubkey, privkey, alpha,
beta)
```

2) Source Code Hash Pesan

Berikut merupakan *source code hash* pesan.

Kode Program *Hash* Pesan

```
def hash(message, k, pubkey):
    added      =      (pubkey[2]**k)%
pubkey[0]

    # Hash
    message = input('Masukkan pesan:
')
    message.append(str(added))

    hashed = 0
    for i in message:
        hashed += ord(i)

    hashed = hashed%277 + 1

    return hashed
```

3) Source Code Tanda Tangan Digital

Berikut merupakan *source code* tanda tangan digital.

Kode Program Tanda Tangan Digital

```
p = int(input('Masukkan nilai p: '))
q = int(input('Masukkan nilai q: '))
a = int(input('Masukkan nilai a: '))

message = input ('Masukkan message:
')
k = int(input('Masukkan nilai k: '))
start_time = time.time()
key = generatekey(p,q,a)
pubkey = key[0]
privkey = key[1]
alfa = int(key[2])
beta = int(key[3])

print('alfa:', alfa)
print('beta:', beta)

hashed = hash(message, k, pubkey)

#Sign
gama = hashed
delta = k + a*(gama%q)
sign = (gama,delta)

print('Tanda tangan digital:', sign)
print('Execution time:')
print("---   %s   seconds   ---"   %
(time.time() - start_time))
```

4) Source Code Verifikasi Tanda Tangan Digital

Berikut merupakan *source code* verifikasi tanda tangan digital.

Kode Program Verifikasi Tanda Tangan Digital

```
p = int(input('Masukkan nilai p: '))
q = int(input('Masukkan nilai q: '))
a = int(input('Masukkan nilai a: '))
message = input('Masukkan pesan: ')
gama   =   int(input('Masukkan   nilai
gama: '))
delta   =   int(input('Masukkan   nilai
delta: '))
k = int(input('Masukkan nilai k: '))

key = generatekey(p,q,a)
pubkey = key[0]
privkey = key[1]
alfa = int(key[2])
beta = int(key[3])
temp   =   (pow(alfa,   delta,   p)   *
pow(beta, -1*gama, p))%p
addedmessage = message + str(temp)

if    (temp    ==    (alfa**k)%p    and
hash(addedmessage,   k,   pubkey)   ==
gama):
    print ('Terverifikasi')
else:
    print ('Tidak terverifikasi')
print('Execution time:')
print("---   %s   seconds   ---"   %
(time.time() - start_time))
```

V. ANALISIS

Penulis melakukan eksperimen yaitu melakukan tanda tangan digital dengan skema Schnorr dan dengan program tanda tangan sebelumnya, menggunakan RSA. Eksperimen dilakukan dengan perangkat keras dan perangkat lunak sebagai berikut:

1. *Operating System* : Windows 10
2. *Processor* : AMD Ryzen 5 4500U

3. *RAM* : 8.00 GB
4. *System type* : 64-bit *operating system,* x64-*based processor*

Eksperimen dilakukan sebanyak 3 dokumen untuk setiap jenis skema tanda tangan digital. Hasil eksperimen diambil dari rata-rata ketiga proses tanda tangan untuk mendapatkan durasi yang objektif. Berikut hasil eksperimen tanda tangan dan verifikasi:

A. Skema Tanda Tangan Digital Schnorr

Coba ke-	Tanda Tangan	Verifikasi
1	0.004083395004272461	0.011994123458862305
2	0.004000663757324219	0.0010035037994384766
3	0.003010272979736328	0.0019865036010742188
Rata-rata	0.003698111	0.00499471

B. Skema Tanda Tangan Digital dengan SHA-1 dan RSA

Coba ke-	Tanda Tangan	Verifikasi
1	0.0019998550415039062	0.003998517990112305
2	0.002000570297241211	0.005003452301025391
3	0.002980470657348633	0.005001544952392578
Rata-rata	0.002326965	0.004667838

Berdasarkan hasil eksperimen, baik tanda tangan maupun verifikasi pada skema Schnorr lebih lambat dibandingkan dengan SHA-1 dan RSA. Hal ini menunjukkan hipotesis awal tidak terbukti. Terdapat alasan ketidaksesuaian hipotesis dan hasil eksperimen, salah satunya adalah penggunaan fungsi *hash* yang berbeda. Fungsi *hash* yang digunakan pada eksperimen ini menggunakan fungsi *hash* dengan perhitungan iteratif sehingga dapat mengonsumsi lebih banyak waktu. Selain itu, kombinasi *hash* dan algoritma tanda tangan digital juga berpengaruh.

VI. Simpulan dan Saran

Dari hasil analisis dan eksperimen yang telah dilakukan, dapat disimpulkan bahwa performa skema tanda tangan digital Schnorr tidak lebih cepat dari skema lain. Hal ini dapat disebabkan karena penggunaan fungsi *hash* yang berbeda. Selain itu, kombinasi *hash* dan algoritma tanda tangan digital juga berpengaruh pada performa skema tanda tangan digital.

Saran untuk penelitian selanjutnya adalah melakukan eksperimen dengan *hash* yang sama yaitu SHA-1 sehingga performa tiap skema dapat dibandingkan lebih setara.

Penghargaan

Puji syukur penulis panjatkan kepada Tuhan Yang Maha Esa karena atas berkat-Nya, penulis dapat menyelesaikan makalah ini. Saya juga berterima kasih kepada Pak Rinaldi Munir selaku dosen mata kuliah II4031 Kriptografi dan Koding STI yang telah membimbing penulis mempelajari ilmu terkait kriptografi selama satu semester. Tak lupa juga saya mengucapkan terima kasih kepada orang tua dan teman-teman yang sudah menemani dan membantu proses belajar penulis di kelas Kriptografi dan Koding.

References

[1] Munir, R. "Bahan Kuliah II4031 Kriptogradi dan Koding", Program Studi Sistem dan Teknologi Informasi ITB, 2021.

[2] S. Huda, A. Sudarsono, and M. Yuliana, "Implementasi Sistem Pengamanan E-Commerce menggunakan Schnorr Digital Signature," *2014 IDSECCONF (Indonesia IT Securiy Conference)*, vol. 1024, pp. 1–9, Oct. 2014.

[3] N. Landman, C. Williams, and E. Ross, "Secure hash algorithms," *Brilliant Math & Science Wiki.* [Online]. Available: https://brilliant.org/wiki/secure-hashing-algorithms/#:~:text=SHA%2D1%20works%20by%20feeding,encrypt%20messages%20using%20SHA%2D1. [Accessed: 25-May-2022].

[4] I. Silaban, P. S. Ramadhan, and D. H. Pane, "Implementasi algoritma schnorr untuk tanda tangan digital Pada Surat pendaftaran online PKBM Hanuba medan," *J-SISKO TECH (Jurnal Teknologi Sistem Informasi dan Sistem Komputer TGD)*, vol. 5, no. 1, p. 25, 2022.

[5] Presiden Republik Indonesia, "PERATURAN PEMERINTAH REPUBLIK INDONESIA NOMOR 82 TAHUN 2012 TENTANG PENYELENGGARAAN SISTEM DAN TRANSAKSI ELEKTRONIK," *JDIH KEMKOMINFO*, 2012. [Online]. Available: https://jdih.kominfo.go.id/produk_hukum/unduh/id/6/t/peraturan+pemerintah+republik+indonesia+nomor+82+tahun+2012. [Accessed: 17-May-2022].

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, 25 Mei 2022

Chintya Wijaya (18219082)