Penggunaan Tiger Hash pada Tanda Tangan Digital

Madiha Ainayya Faizzati (18218010)
Program Studi Sistem dan Teknologi Informasi
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung, Jalan Ganesha 10 Bandung
E-mail : madiha.ainayya@gmail.com

***Abstract*—Meningkatnya *digital document sharing* utamanya dokumen penting melalui internet akibat pandemi COVID-19 membuat adanya kebutuhan akan mekanisme untuk menjamin keaslian dokumen dan autentikasi identitas pengirim dokumen. Tanda tangan digital merupakan metode yang dapat memenuhi kebutuhan tersebut. Melalui tanda tangan digital yang dibubuhkan pada sebuah dokumen, penerima dapat memeriksa identitas pengirim dan keaslian dokumen yang diterimanya. Salah satu langkah dari pembuatan dan verifikasi tanda tangan digital adalah implementasi fungsi *hash*. Fungsi *hash* yang paling umum digunakan dalam tanda tangan digital adalah SHA-1, SHA-256, serta MD5 [7]. Oleh karena itu, penulis ingin mempelajari implementasi fungsi *hash* lainnya yaitu Tiger *hash* dalam tanda tangan digital. Dari implementasi tersebut, dapat dibandingkan performa Tiger dibanding fungsi-fungsi *hash* yang lebih populer sehingga dapat ditarik sebuah kesimpulan mengenai alasan ketiga fungsi *hash* tersebut lebih umum digunakan pada tanda tangan digital daripada Tiger.**

Keywords—tanda tangan digital, Tiger, performa

I. PENDAHULUAN

Di era serba digital, keamanan informasi menjadi salah satu hal yang menjadi perhatian pengguna internet. Berbagai upaya dari pelaku kejahatan siber untuk merusak, memanipulasi, mencuri, dan menyebarkan informasi sensitif milik individu maupun badan tertentu pun semakin meningkat. Menurut Badan Siber dan Sandi Negara [5], dari Januari – April 2020, tercatat 88.414.296 serangan siber. Hal ini membuktikan bahwa pengguna internet memiliki urgensi tinggi untuk menjamin *information sharing* yang dilakukan melalui *platform* digital terjamin keamanannya. Ditambah dengan pandemi COVID-19, angka *work from home* (WFH) diprediksi akan meningkat sebanyak 25-30% pada akhir tahun 2021 [6]. Dengan tren tersebut, tentunyan kegiatan *document sharing* secara daring melalui kanal internet juga akan meningkat; tak terlepas *document sharing* untuk dokumen-dokumen penting yang memiliki nilai tinggi.

Sehingga, diperlukan mekanisme untuk menjamin keaslian suatu dokumen dan mekanisme autentikasi pengirim dokumen tersebut. Tanda tangan digital merupakan salah satu solusi yang bisa menjamin *document sharing* lebih tepercaya dan aman. Ada banyak aspek yang bisa ditinjau dari tanda tangan digital; salah satunya adalah fungsi *hash* yang digunakan. Pada penelitian ini, fungsi *hash* yang tidak umum digunakan untuk tanda tangan digital yaitu Tiger *hash* akan diimplementasikan dan dibandingkan performanya dengan fungsi *hash* lainnya yang lebih umum digunakan. Performa yang dimaksud adalah kecepatan eksekusi tahapan-tahapan *signing* dan verifikasi dari tanda tangan digital.

II. METODE

A. Literature Review

Pertama-tama, akan dilakukan *literature review* dengan berbagai sumber cetak maupun digital mengenai konsep tanda tangan digital dan Tiger Hash. Dari *literature review*, akan ditemukan informasi mengenai performa Tiger Hash dibandingkan dengan algoritma Hash lainnya yaitu SHA-1, SHA-256, dan MD5.

B. Eksperimen

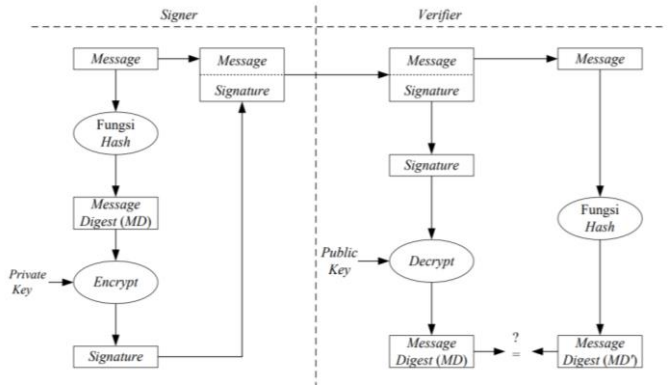
Selanjutnya, akan dibuat program tanda tangan digital dengan mengimplementasikan Tiger Hash dan RSA dengan bahasa pemrograman Python. Dengan program ini, akan dilihat secara langsung dan eksperimental bagaimana implementasi dan performa Tiger Hash pada tanda tangan digital.

Sebagai pembanding, akan diimplementasikan pula tanda tangan digital menggunakan RSA dan SHA-1, SHA-256, dan MD5. Semua program akan ditulis dengan bahasa Python dan menggunakan pustaka kriptografi yang sudah tersedia secara publik.

III. TANDA TANGAN DIGITAL PADA DOKUMEN

Tanda tangan digital merupakan skema matematis yang digunakan untuk menjamin keaslian pesan ataupun dokumen digital [1]. Seperti namanya, tanda tangan digital memiliki konsep yang mirip dengan tanda tangan tradisional. Keduanya sama-sama memiliki karakteristik unik untuk setiap individu sehingga penandatangan dapat diidentifikasi berdasarkan tanda tangan yang dibubuhkan. Selain sebagai mekanisme autentikasi penandatangan, tanda tangan digital juga dapat digunakan sebagai indikator keaslian dokumen; apakah dokumen atau pesan yang diterima sudah diubah/dimanipulasi oleh *unauthorized personnel* selama masa pengirimannya atau tidak [2].

Tanda tangan digital menggunakan fungsi *hash* daninfrastruktur kunci publik dalam operasinya, sesuai dengan



Gambar 1
(sumber:
https://informatika.stei.itb.ac.id/~rinaldi.munir/Kriptografi-dan-Koding/Tanda-tangan-digital-2021.pdf)

diagram alir pada Gambar 1. Terdapat dua aktor yang terlibat yaitu pengirim pesan (*signer*) dan penerima pesan (*verifier*).

Mula-mula, pengirim pesan akan melakukan fungsi *hash* terhadap pesan yang ingin dikirimkannya. Operasi *hashing* tersebut menghasilkan *message digest* yang kemudian dienkripsi dengan kunci privat milik pengirim. Hasil enkripsi tersebutlah yang menjadi tanda tangan digital yang disisipkan ke dalam pesan. Agar tanda tangan dapat dibedakan dari pesan, tanda tangan tersebut diberi penanda khusus yang disepakati oleh penerima dan pengirim, contohnya diberi *tag* <signature></signature>. Setelah itu, pesan yang sudah dibubuhi tanda tangan digital dikirim melalui kanal digital.

Di sisi penerima, mula-mula penerima menerima pesan dari pengirim. Dari pesan tersebut, penerima akan mengekstrak pesan dan tanda tangan digital. Pesan dimasukkan ke dalam fungsi *hash* yang kemudian menghasilkan *message digest*. Sementara itu, tanda tangan digital didekripsi menggunakan kunci publik pengirim dan dihasilkan pula *message digest*. Kedua *message digest* tersebut kemudian dicek kesamaannya. Apabila sama, artinya pengirim terautentikasi dan pesan terbukti asli.

Secara praktik, tanda tangan digital dapat digunakan untuk berbagai dokuemen digital: email/surel, transaksi kartu kredit, ataupun dokumen digital lainnya. Dokumen tersebut bisa memililiki fungsi, tujuan, dan konten apa saja, namun dokumen yang paling umum diberi pengaman berupa tanda tangan digital adalah dokumen penting yang membutuhkan autentikasi identitas pengirim, seperti dokumen perjanjian dan transkasi elektronik [12]. Pemberian tanda tangan digital memberikan kekuatan hukum dan akibat hukum yang sah bagi dokumen yang ditandatangani.

Di Indonesia sendiri, perihal tanda tangan digital diatur pada Peraturan Pemerintah tentang Penyelenggaraan Sistem dan Transaksi Elektronik. Berlandaskan dasar hukum tersebut, [13], dijelaskan pula bahwa dokumen digital yang dapat diberi tanda tangan digital sebagai mekanisme keamanan memiliki bentuk yang tidak terbatas: tulisan, suara, gambar, peta, rancangan, foto atau sejenisnya, huruf, tanda, dan sebagainya yang memiliki makna atau arti oleh orang yang mampu memahaminya.

Dari situ, dapat dilihat bahwa tanda tangan digital pada dokumen bukanlah hanya sebuah metode keamanan saja, tapi sudah menjadi landasan sah secara hukum atau tidaknya suatu perjanjian. Dengan peran tanda tangan digital besar tersebut dalam dunia digital, topik mengenai tanda tangan digital dan mekanisme-mekansisme optimasinya menarik untuk dipelajari.

IV. TIGER HASH

Tiger adalah fungsi kriptografi *hash compression* satu arah yang didesain oleh Ross Anderson dan Eli Biham pada tahun 1996 [3]. Tiger *hash* memproses pesan dalam bentuk blok-blok berukuran 512 bit menjadi *message digest* berukuran 192 bit, meskipun ada juga versi Tiger/160 dan Tiger/128 yang masing-masing menghasilkan *message digest* dengan ukuran 160 bit dan 128 bit. Pada [4] yang dirilis oleh Anderson dan Biham, mereka menyatakan bahwa Tiger memiliki kecepatan sama dengan SHA-1 pada *processor* 32 bit dan tiga kali lebih cepat pada *processor* 64 bit.

Secara umum, fungsi Tiger memiliki dua bagian: *state update transformation* dan *key schedule*. *Hashing* dimulai dengan inisialisasi *fixed initial value* ke dalam 3 register a, b, dan c yang masing-masing berukuran 64 bit. Sementara itu, satu blok pesan berukuran 512 bit dibagi menjadi 8 buah blok berukuran 64 bit x0, x1, …, x7. Nilai a, b, dan c disimpan sebagai *state variable* dan akan di-*update* dengan 3 *pass* yang masing-masing memiliki 8 *round*. Bagian tersebutlah yang disebut sebagai *state update transform*.

Definisi dari operasi *pass* dan *round* adalah sebagai berikut.

```
pass(a, b, c, mul) adalah
          round(a, b, c, x0, mul);
          round(b, c, a, x1, mul);
          round(c, a, b, x2, mul);
          round(a, b, c, x3, mul);
          round(b, c, a, x4, mul);
          round(c, a, b, x5, mul);
          round(a, b, c, x6, mul);
          round(b, c, a, x7, mul);
```

```
round(a, b, c, x, mul) adalah
c ^= x;
a -= t1[c_0] ^ t2[c_2] ^ t3[c_4] ^ t4[c_6];
b += t4[c_1] ^ t3[c_3] ^ t2[c_5] ^ t1[c_7];
b *= mul;
```

dengan c_i mendefinisikan bit ke-i dari c. t1, t2, t3, dan t4 adalah S-*box* milik Tiger.

Sementara itu, bagian *key schedule* adalah sebagai berikut.

```
x0 -= x7 ^ 0xA5A5A5A5A5A5A5A5;
x1 ^= x0;
x2 += x1;
x3 -= x2 ^ ((~x1)<<19);
x4 ^= x3;
x5 += x4;
x6 -= x5 ^ ((~x4)>>23);
x7 ^= x6;
x0 += x7;
x1 += x0 ^ ((~x7)<<19);
x2 ^= x1;
x3 += x2;
x4 -= x3 ^ ((~x2)>>23);
x5 ^= x4;
x6 += x5;
x7 -= x6 ^ 0x0123456789ABCDEF;
```

Selain *state update transformation* dan *key schedule*, terdapat dua operasi minor untuk menyimpan dan mengeksekusi operasi matematis terhadap *fixed initial value*; sehingga meskipun nilai register a, b, dan c akan selalu di-*update* pada setiap *pass*, nilai awal tetap tesimpan karena akan digunakan kembali di akhir proses *hashing*.

`save_abc` adalah

```
aa = a;
bb = b;
cc = c;
```

`feedforward` adalah

```
a ^ = aa;
b -= bb;
c += cc;
```

Berdasar definisi-definisi operasi di atas, algoritma *compression* Tiger adalah sebagai berikut.

```
save_abc;
pass(a, b, c, 5);
key_schedule;
pass(c, a, b, 7);
key_schedule;
pass(b, c, a, 9);
feedforward;
```

Penggunaan Tiger *hash* sendiri masih belum umum, karena literatur yang membicarakan implementasinya pada suatu skema kriptografi tertentu juga tidak banyak ditemukan. Namun, salah satunya adalah TTH (Tiger Tree Hash) Root yang digunakan pada server *file sharing* DirectConnect, dan digunakan pula sebagai *hash* sekunder pada jaringan GNUTella [14].

Dari segi aspek keamanan, kriptanalis telah mencoba menemukan *vulnerability* dari Tiger. Hasilnya, ditemukan bahwa telah ditemukan 1 bit *circular pseudo-near collision* pada fungsi *hash* Tiger penuh (24 *round*). *Collision attack* adalah ketika *attacker* mampu menemukan dua masukan yang dapat menghasilkan nilai *hash* yang sama. Hal ini sebenarnya cukup baik jika dibandingkan dengan MD5 dan SHA-1 yang diketahui memiliki banyak kasus *collision attack*.

Namun, informasi keamanan mengenai Tiger ini bisa jadi *biased* karena jika dibanding dengan MD5 dan SHA-1, Tiger memang kurang populer. Sehingga, bisa jadi terdapat *vulnerability* lain pada Tiger yang belum terungkap akibat minimnya inseitf dan kompetisi untuk melakukan *attack* pada Tiger.

V. SHA-1, SHA-256, DAN MD5

Selain Tiger *hash*, fungsi *hash* lain yang perlu dikenal adalah SHA-1, SHA0256, dan MD5 yang akan dibandingkan dengan Tiger. Meski begitu, pembahasan mengenai konsep masing-masing fungsi *hash* tidak akan sedalam Tiger dan hanya singkat saja.

A. SHA-1

SHA-1 (Secure Hash Algorithm 1) adalah salah satu fungsi *hash* dengan nilai *hash* yang dihasilkan 160 bit. SHA-1 dikembangkan pada tahun 1993 oleh United States National Security Agency (NIST) [15]. SHA-1 sudah dianggap aman sejak tahun 2005. *Vulnerability* yang ditemukan pada SHA-1 adalah *collision attack*, namun dengan kecepatan penemuan *collision* yang tinggi. Oleh karena itu, banyak perusahaan teknologi besar yang kemudian menghentikan penggunaan SHA-1 dari sistem mereka demi menjaga keamanan data mereka.

B. SHA-256

Selanjutnya, karena masalah keamanan yang ditemukan pada SHA-1, dikembangkanlah SHA-2 oleh United States National Security Agency (NSA) pada tahun 2001. SHA-2 berbeda secara signifikan dari pendahulunya yaitu SHA-1. SHA-2 bisa disebut sebagai ‘keluarga’ karena memiliki fungsi

hash turunannya yang memiliki nilai *hash* yang berbeda-beda; SHA-256 adalah keluarga SHA-2 yang memiliki nilai *hash* 256. Saat ini, SHA-256 adalah satu satu fungsi *hash* yang masih dianggap paling aman, terbukti hingga saat ini pemerintah Amerika Serikat masih menggunakan SHA-256 pada standar keamanannya.

C. MD5

MD5 adalah fungsi *hash* yang memiliki nilai *hash* 128 bit. MD5 dikembangkan pada tahun 1991 oleh Ronald Rivest dan memiliki pendahulu MD5. Namun, tidak lama kemudian, yaitu pada tahun 1996, ditemukan bahwa MD5 memiliki *vulnerability* terhadap *collision attack* sehingga sudah dianggap tidak aman. Meski begitu, hingga saat ini, MD5 masih cukup digunakan pada beberapa kasus.

VI. Desain Program

Implementasi tanda tangan digital menggunakan Tiger dan fungsi *hash* lain sebagai pembanding akan dilakukan menggunakan bahasa pemrograman Python. Python dipilih karena *library* kriptografi yang dimiliki cukup lengkap dan terdapat banyak dokumentasi mengenai implementasi algoritma kriptografi menggunakan Python.

Library dan *source* yang digunakan dalam implementasi program tanda tangan digital adalah sebagai berikut.

1. RSA menggunakan modifikasi program implementasi tugas kecil mata kuliah II4031 Kriptografi & Koding milik kelompok penulis [8]
2. Tiger menggunakan program implementasi milik dtitenko-dev [9]
3. Fungsi *hash* SHA-1, SHA-256, dan MD5 menggunakan *library* hashlib [10]

Sebenarnya, Python sudah memiliki *library* Tiger sendiri yang dirilis pada PyPI [11]. Namun, *library* tersebut sudah *deprecated* karena dirilis pada tahun 2010 dan memiliki *requirement* Python 2.4 yang sudah cukup tertinggal dan *maintenance*-nya pun sudah dihentikan. Oleh karena itu, digunakan alternatif lain untuk implementasi Tiger dengan menggunakan *source* terbaru yang masih bisa dijalankan pada Python 3 ke atas.

Fenomena sulitnya mencari *library* Tiger terbaru yang *compatible* dengan perangkat saat ini cukup menarik. Sebab, artinya komunitas pengembang tidak menemukan urgensi untuk terus melakukan *maintenance library* Tiger; hal tersebut mengkonfirmasi penggunaan Tiger yang tidak umum.

Berikut adalah spesifikasi program tanda tangan digital yang dibuat.

1. Dokumen akan dianalogikan sebagai data bertipe String yang memiliki nilai tetap, sehingga pesan yang ditandatangani tidak benar-benar berbentuk dokumen
2. Nilai p dan q untuk RSA memiliki nilai tetap
3. Program dieksekusi dari terminal (tanpa *graphical user interface*)
4. Program dilengkapi dengan *timer* untuk melihat kecepatan proses *signing* dan verifikasi

Sehingga, pada akhir pengkodean, akan ada empat *file* Python (tidak termasuk *file dependencies*) untuk empat tanda tangan digital berbeda: menggunakan Tiger, SHA-1, SHA-256, dan MD5.

VII. Hasil Eksperimen dan Pembahasan

Setelah kode selesai ditulis, penulis melakukan eksperimen secara individu pada tanggal 24 Mei 2021. Eksperimen dilakukan menggunakan perangkat keras dan lunak dengan spesifikasi sebagai berikut.

1. *Operating system*: Windows 10
2. *Processor*: Intel® Core™ i7-8550U CPU @ 1.80GHz 1.99 GHz
3. RAM: 8,00 GB
4. *System type*: 64-bit *operating system*, x64-*based processor*

Eksperimen untuk setiap jenis tanda tangan digital dilakukan sebanyak tiga kali, sehingga akan ada tiga pasang nilai durasi *signing* dan verifikasi. Selanjutnya, ketiga nilai *signing* dan verifikasi dirata-rata untuk mendapatkan durasi yang lebih objektif. Berikut durasi *signing* dan verifikasi dari empat program tanda tangan digital. Durasi yang ditulis memiliki satuan sekon.

Tabel 1 Hasil Tiger

Try	***Signing***	**Verifikasi**
1	0.00049070	0.00518680
2	0.00205150	0.00798460
3	0.00307950	0.01255700
Rata-rata	0.0018739	0.008576133333

Tabel 2 Hasil SHA-1

Try	***Signing***	**Verifikasi**
1	0.00035290	0.00704320
2	0.00585920	0.00717030
3	0.00152440	0.00846890
Rata-rata	0.002578833333	0.0075608

Tabel 3 Hasil SHA-256

Try	***Signing***	**Verifikasi**
1	0.00056390	0.00459840
2	0.00156970	0.00896540
3	0.00248150	0.00943620
Rata-rata	0.001538366667	0.007666666667

Tabel 4 Hasil MD5

Try	***Signing***	**Verifikasi**
1	0.00146910	0.00765290
2	0.00147250	0.00803730
3	0.00151780	0.00886070
Rata-rata	0.001486466667	0.008183633333

Rata-rata durasi *signing* dan verifikasi dari keempat jenis tanda tangan digital di-*plot* menjadi grafik *bar chart* pada Gambar 2 agar dapat dibandingkan dengan lebih mudah. Dapat dilihat bahwa secara umum, proses verifikasi memiliki durasi yang jauh lebih lama daripada proses *signing*. Selain

itu, pendapat Anderson dan Biham mengenai Tiger yang tiga kali lebih cepat dari SHA-1 pada *processor* 64 bit tidak terbukti. Meskipun durasi *signing* milik Tiger memang lebih cepat daripada SHA-1, selisihnya tidak cukup besar. Terlebih, pada verifikasi, Tiger justru memiliki durasi yang lebih lama daripada SHA-1. Bahkan, dibanding tanda tangan digital dengan fungsi *hash* lainnya pun, Tiger memiliki durasi verifikasi yang paling lama.

Hal ini bisa jadi disebabkan oleh adanya implementasi kode yang kurang efisien. Kejadian ini sangat mungkin terjadi mengingat *source code* Tiger yang digunakan bukan berasal dari *standard library* Python yang di-*maintain* secara rutin. Apalagi jika dibandingkan dengan fungsi *hash* SHA-1, SHA-256, dan MD5 yang diimplementasikan dengan *library* hashlib yang merupakan *standard library* Python yang tentunya sudah melalui berbagai proses *quality assurance* dan *maintenance* untuk terus mengoptimalisasi performanya. Kode yang tidak efisien ini bisa memiliki berbagai bentuk: bisa jadi struktur data yang digunakan maupun pemilihan *syntax* yang digunakan masing-masing pembuat *library*/*source* tidak optimum.

Hipotesis lain dari mengapa performa Tiger tidak seperti klaim adalah ada kemungkinan performa *hash* secara individu akan berbeda dari performa *hash* jika diimplementasikan bersama algoritma kriptografi lain, dalam hal ini RSA, untuk membangun suatu program tanda tangan digital. Bisa jadi jika diuji secara individu, performa Tiger baik, namun tidak *compatible* dengan implementasi tanda tangan digital.

Gambar 2 Perbandingan Durasi *Signing* dan Verifikasi Empat Tanda Tangan Digital dengan Fungsi *Hash* Berbeda
(sumber: dokumentasi penulis)

VIII. KESIMPULAN

Dari hasil analisis dan eksperimen yang dilakukan, dapat disimpulkan bahwa pada implementasi Tiger untuk tanda tangan digital menggunakan bahasa Python, performa dari tanda tangan digital Tiger tidak lebih baik dari performa tanda tangan digital yang menggunakan fungsi *hash* lain yaitu SHA-1, SHA-256, dan MD5. Hal ini bisa menjadi penjelasan mengapa fungsi *hash* Tiger tidak umum digunakan untuk implementasi tanda tangan digital, meskipun secara teoretis, seharusnya keamanan Tiger lebih baik daripada SHA-1 dan MD5 yang sudah dianggap tidak aman karena *exposure* terhadap *collision attack* yang cukup tinggi.

PENGHARGAAN

Saya mengucapkan terima kasih kepada Tuhan Yang Maha Esa atas berkatNya, kep ada orang tua saya yang tak henti memberikan dukungan kepada saya, dan kepada Pak Rinaldi Munir selaku dosen mata kuliah Kriptografi dan Koding yang telah membimbing saya dalam belajar ilmu terkait kriptografi. Selain itu, saya juga mengucapkan terima kasih kepada teman-teman Sistem dan Teknologi Informasi yang telah menemani dan membantu proses belajar saya di kelas Kriptografi dan Koding: Tafia, Ilman, Nisa, dan Adi.

REFERENSI

[1] EMP Trust HR. (2017, Sept. 12). *What are Digital Signatures: How it works, Benefits, Objectives, Concept* [Online]. Available: https://www.emptrust.com/blog/benefits-of-using-digital-signatures

[2] CloudHost. (2020, Des. 22). *Digital Signature (Tanda Tangan Digital) : Pengertian, Fungsi, Cara Kerja, dan Keunggulannya* [Online]. Available: https://idcloudhost.com/digital-signature-tanda-tangan-digital-pengertian-fungsi-cara-kerja-dan-keunggulannya/

[3] F. Mendel and V. Rijmen, "Cryptanalysis of the Tiger Hash Function," International Association for Cryptologic Research [Online]. Available: https://iacr.org/archive/asiacrypt2007/48330539/48330539.pdf

[4] R Anderson and E. Biham.

[5] Badan Siber dan Sandi Negara. (2020, Apr. 2020). *Rekap Serangan Siber (Januari – April 2020)* [Online]. Available: https://bssn.go.id/rekap-serangan-siber-januari-april-2020/

[6] Glonal Workplace Analytics. (n.d.). *Work-At-Home After Covid-19—Our Forecast* [Online]. Available: https://globalworkplaceanalytics.com/work-at-home-after-covid-19-our-forecast

[7] US Department of Homeland Security CISA Cyber + Infrastructure. (2020, Aug. 20). *Understanding Digital Signatures* [Online]. Available: https://us-cert.cisa.gov/ncas/tips/ST04-018

[8] milmanss. (2021, Apr. 23). *ii4031_tugaskecil_4* [Online]. Available: https://github.com/milmanss/ii4031_tugaskecil_4

[9] dtitenko-dev. (2016, Dec. 21). *tigerhash* [Online]. Available: https://github.com/dtitenko-dev/tigerhash

[10] Python. (n.d.). *hashlib — Secure hashes and message digests* [Online]. Available: https://docs.python.org/3/library/hashlib.html

[11] PyPI. (2010, Aug. 16). *tiger 0.3* [Online]. Available: https://pypi.org/project/tiger/

[12] Hukum Online. (2020, May. 5). *Cara Kerja Tanda Tangan Elektronik* [Online]. Available: https://new.hukumonline.com/klinik/detail/ulasan/cl3/cara-kerja-tanda-tangan-elektronik/

[13] Presiden Republik Indonesia. (n.d.) *PERATURAN PEMERINTAH REPUBLIK INDONESIA NOMOR 82 TAHUN 2012 TENTANG PENYELENGGARAAN SISTEM DAN TRANSAKSI ELEKTRONIK* [Online]. Available: https://jdih.kominfo.go.id/produk_hukum/unduh/id/6/t/peraturan+pemerintah+republik+indonesia+nomor+82+tahun+2012

[14] Feit, Harold. (2012, Feb 12). *P2P:Protocol:Specifications:Optional Hashes* [Online]. Available: http://wiki.depthstrike.com/P2P:Protocol:Specifications:Optional_Hashes#TTH_Root

[15] Scheiner. (n.d.). *Cryptanalysis of SHA-1* [Online]. Available: https://www.schneier.com/blog/archives/2005/02/cryptanalysis_o.html

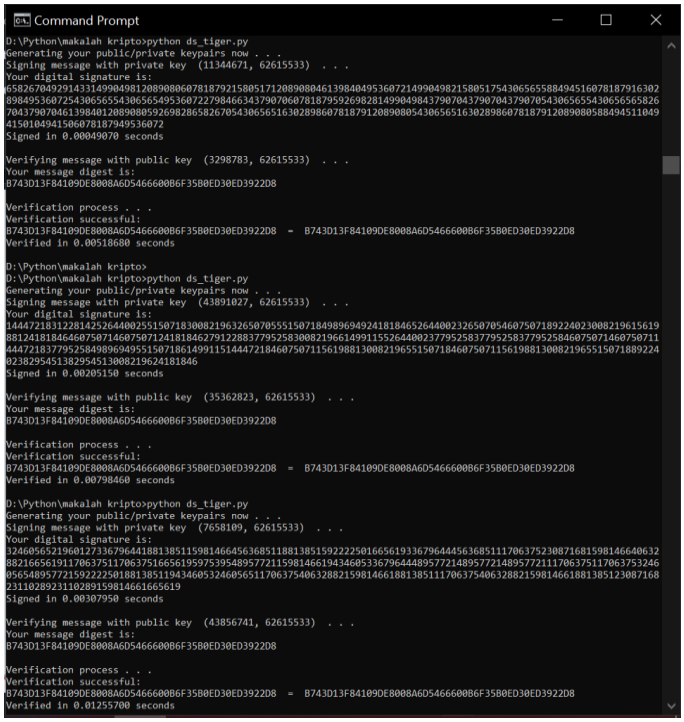
PERNYATAAN

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.

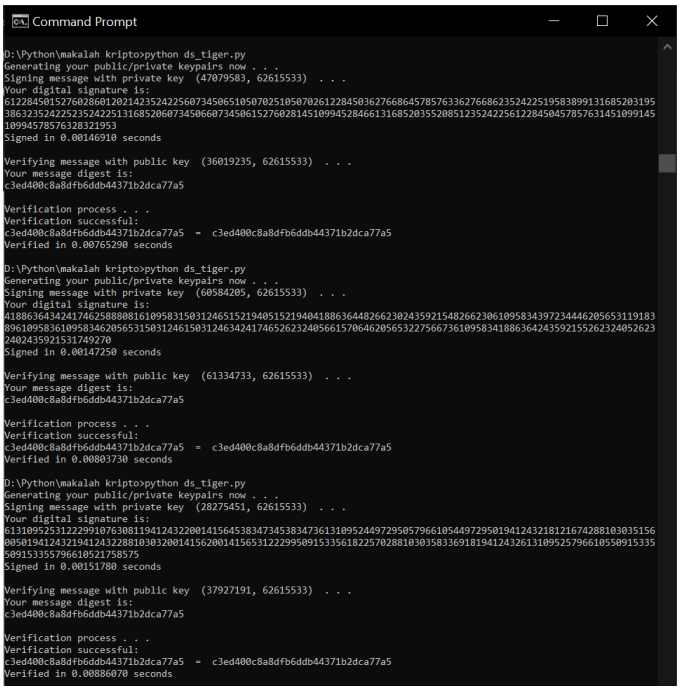
Bandung, 25 Mei 2021

Madiha Ainayya Faizzati (18218010)

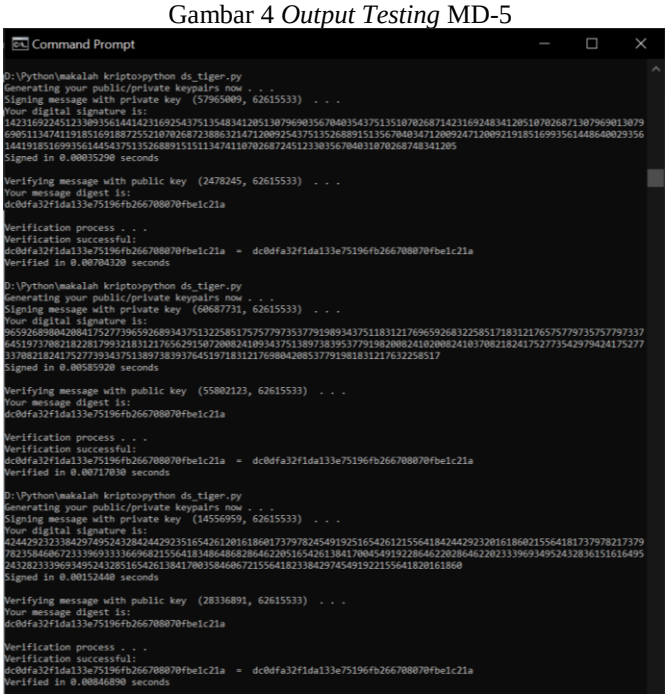
APPENDIX



Gambar 3 *Output Testing* Tiger Hash



Gambar 4 *Output Testing* MD-5



Gambar 5 *Output Testing* SHA-1

Gambar 6 *Output Testing* SHA-256