# Analisis Integritas Dokumen Digital Pada Aplikasi Digisign UTD PNJ Menggunakan Tanda Tangan Digital

**Muhammad Dimas Yudha Adhi Pratama Jr.** [1], **Defiana Arnaldy, S.TP, M.Si** [2]

Program Studi Teknik Multimedia dan Jaringan, Jurusan Teknik Informatika dan Komputer,
Politeknik Negeri Jakarta
Depok, Indonesia

muhammad.dimasyudhaadhipratama.tik18@mhsw.pnj.ac.id[1] , defiana.arnaldy@tik.pnj.ac.id[2]

**Abstrak**

Dewasa ini, penggunaan dokumen dalam bentuk digital atau dokumen elektronik semakin sering digunakan. Dikarenakan dokumen tersebut bersifat digital, akan sangat mudah untuk dilakukan modifikasi oleh segala pihak yang memiliki dokumen tersebut. Pada saat ini Politeknik Negeri Jakarta sedang membangun aplikasi Digisign yang digunakan untuk membuat dan memverifikasi dokumen digital dengan menerapkan teknologi tandatangan digital. Dalam *Digital Signature Standard* yang diterbitkan NIST, terdapat beberapa algoritma penandatanganan yang memenuhi standar tersebut, seperti DSA, RSA, dan ECDSA. Setiap algoritma memiliki kelebihan dan kekurangan masing-masing, maka dari itu perlu dikaji lebih lanjut dari ketiga algoritma yang merupakan algoritma terbaik untuk aplikasi Digisign dengan melakukan pengujian performa setiap algoritma pada aplikasi Digisign. Parameter pada pengujian ini meliputi penggunaan waktu penggunaan CPU, dan penggunaan *memory* pada tiga proses yang dilakukan pada aplikasi Digisign, yaitu pembuatan pasangan kunci, penandatanganan dokumen, dan verifikasi dokumen. Dalam pengujiannya digunakan 3 buah sampel dalam bentuk skema antara lain dokumen yang hanya berisi teks, dokumen campuran teks dan gambar, dan dokumen yang telah memiliki tandatangan. Hasil dari penelitian ini menunjukkan algoritma ECDSA lebih unggul dalam penggunaan waktu yang lebih cepat dibanding dua algoritma lainnya, sedangkan DSA lebih unggul dalam penggunaan CPU yang lebih sedikit, serta RSA yang lebih unggul dalam penggunaan *memory*.

**Kata kunci:** Algoritma tandatangan digital, DSA, RSA, ECDSA, Performa algoritma

## 1. PENDAHULUAN

Dewasa ini, penggunaan dokumen dalam bentuk digital atau dokumen elektronik semakin sering digunakan. Dikarenakan dokumen tersebut bersifat digital, akan sangat mudah untuk dilakukan modifikasi oleh segala pihak yang memiliki dokumen tersebut. Pada saat ini Politeknik Negeri Jakarta sedang membangun aplikasi Digisign yang digunakan untuk membuat dan memverifikasi dokumen digital dengan menerapkan teknologi tanda tangan digital.

Tanda tangan digital adalah salah satu teknologi yang dapat digunakan untuk melakukan pembuktian secara matematis bahwa data tidak mengalami modifikasi secara ilegal, sehingga bisa digunakan sebagai salah satu solusi untuk melakukan pemeriksaan integritas data dokumen yang sah (Finandhita dan Afrianto, 2018) (Nugraha, 2017). Namun, kekuatan dan ketahanan dari tanda tangan digital sangat bergantung dengan metode kriptografi dan panjang kunci yang digunakan (Afrianto et al., 2020).

Dalam Digital Signature Standard yang diterbitkan NIST, terdapat beberapa algoritma penandatanganan yang memenuhi standar tersebut, seperti DSA, RSA, dan ECDSA. Setiap algoritma memiliki kelebihan dan kekurangan masing-masing, maka dari itu perlu dikaji lebih lanjut dari ketiga algoritma yang merupakan algoritma terbaik untuk aplikasi Digisign dengan melakukan pengujian performa setiap algoritma pada aplikasi Digisign.

## 2. INTEGRITAS DATA

Dalam keamanan informasi, terdapat beberapa aspek yang perlu diperhatikan. Aspek-aspek ini memiliki sebutan CIA *Triad* yang terdiri dari *confidentiality* (kerahasiaan), *integrity* (integritas), dan *availability* (ketersediaan). Parameter-parameter dari 3 aspek tersebut digunakan untuk pengukuran keamanan dari suatu sistem (Pramono, 2019).

## 3. TANDATANGAN DIGITAL

Tanda tangan digital adalah bentuk analog elektronik dari tanda tangan tertulis atau konvensional. Tanda tangan digital digunakan untuk menjamin bahwa berkas atau dokumen yang ditandatangani adalah benar telah ditandatangani. Fungsi lain dari tanda tangan digital adalah untuk memastikan integritas suatu dokumen atau berkas telah dimodifikasi atau tidak setelah ditandatangani. Tanda tangan digital terdiri dari 2 proses, yaitu pembuatan tanda tangan (*sign*) dan verifikasi tanda tangan.

Pada proses pembuatan tanda tangan data/dokumen akan diambil *message digest* (MD) dengan panjang tetap melalui proses *hashing*. Setelah MD didapatkan, MD dari dokumen akan dilakukan enkripsi menggunakan

algoritma penandatanganan (*Signature Algorithm*) yang dipasangkan dengan kunci privat. Sehingga menghasilkan tanda tangan digital lalu dipasangkan dengan data/dokumen terkait. Perlu diingat dalam setiap penandatanganan setidaknya memiliki sebuah pasangan kunci publik dan privat. Kunci privat digunakan untuk pembuatan tanda tangan dan kunci publik untuk verifikasi tanda tangan.

Dokumen/data yang telah dipasangkan dengan tanda tangan digital akan dikirim kepada penerima. Jika penerima ingin melakukan verifikasi, maka data/dokumen akan dilakukan *hashing* untuk menghasilkan MD. Selanjutnya penerima dapat menggunakan kunci publik pengirim dengan memasukkan ke algoritma verifikasi tanda tangan dengan membandingkan hasil MD dari dokumen yang diterima dengan nilai MD dari tanda tangan yang telah didekripsi menggunakan kunci publik pengirim. Jika MD dari pengirim dan MD dari dokumen yang diterima oleh penerima sama atau sesuai, maka dokumen/data dinyatakan valid atau benar dan berlaku sebaliknya (NIST, 2015).

Pada penelitian ini algoritma yang dibandingkan adalah DSA, RSA, dan ECDSA.

**A. DSA**

DSA digunakan untuk tanda tangan digital, baik menandatangani pesan dan memverifikasi tanda tangan yang menggunakan perhitungan matematika modulus exponensial dan logaritma diskrit (Svetlin Nakov, 2018).

Langkah-langkah utama DSA adalah sebagai berikut:

1. Pilihan parameter algoritma:

Pilih fungsi hash kriptografi H, ukuran kunci L, bilangan prima q dengan jumlah bit yang sama dengan keluaran H, bilangan prima L-bit p sehingga p-1 adalah kelipatan q, bilangan g yang modulus orde perkaliannya p adalah q.

2. Pembuatan Kunci

• Pilih x dengan beberapa metode acak, di mana 0 < x < q .

• Menghitung y = $g^x$ mod p .

• Kunci publik adalah (p, q, g, y). Kunci pribadi adalah x.

3. Pembuatan tanda tangan

• Hasil nilai pesan acak k di mana 0 < k < q .

• Menghitung r = ($g^k$ mod p) mod q .

• Menghitung $s$ = ($k^{-1}$ ($H(m)$ + $x$*$r$)) mod q , Dimana H adalah fungsi hashing dan m adalah pesannya.

• Menghitung ulang tanda tangan dalam kasus yang tidak mungkin terjadi yaitu r=0 atau s=0.

• Tanda tangan adalah (r,s).

4. Verifikasi Tanda tangan

• Tanda tangan ditolak jika 0< r <q atau 0< s <q tidak terpenuhi.

• Menghitung w = $(S)^{-1}$mod q .

• Menghitung u1 = (H(m)*w) mod q .

• Menghitung u2 = (r*w) mod q .

• Menghitung $v$ = (($g^{u1}$ *$y^{u2}$) mod $p$) mod $q$

• Tanda tangan valid jika v = r .

**B. RSA**

RSA sebagai kriptosistem kunci publik menyediakan skema tanda tangan digital yang terdiri dari pembuatan kunci, sign atau penandatanganan, dan verifikasi. Panjang kunci yang dihasilkan sangat beragam dari 128-bit hingga 16384-bit. Namun semakin panjang kunci, akan semakin lambat pula dalam prosesnya (Svetlin Nakov, 2018).

Langkah-langkah utama RSA adalah sebagai berikut:

1. Pembuatan kunci

• Pilih dua bilangan prima yang berbeda p dan q.

• Dilakukan komputasi n = pq dan n digunakan sebagai modulus untuk kunci publik dan kunci privat.

• Dilakukan komputasi totalitas ϕ(n) = ( p −1)(q −1) .

• Pilih bilangan bulat e sehingga 1< e <ϕ(n) , dan e dan (n) tidak berbagi faktor selain 1 (misalnya e dan (n) adalah *coprime*).

• Determinan d (menggunakan aritmatika modular) yang memenuhi hubungan kongruensi *de* α1(mod∏(*n*)) .

Kunci publik terdiri dari modulus n dan eksponen publik e. Kunci privat terdiri dari modulus n dan eksponen privat (atau dekripsi) d yang harus dirahasiakan.

2. Pembuatan tanda tangan

• Menghasilkan nilai hash h1 dari pesan m: h1 = H(m) .

• Menghasilkan tanda tangan S menggunakan kunci privat (d,n) : $s = h_1^d$ (mod $n$).

3. Verifikasi tanda tangan

• Dapatkan kunci publik (e,n) pengirim.

• Dekripsi tanda tangan S menggunakan kunci publik (e,n): $h = S^e$ (mod $n$).

• Menghasilkan nilai hash menggunakan algoritma hash yang sama dalam hubungannya dengan kunci publik pengirim dan membandingkannya dengan hasil h .Jika keduanya setuju, tanda tangan valid, jika tidak, tanda tangan tidak valid.

RSA-PSS adalah skema tanda tangan baru yang didasarkan pada kriptosistem RSA dan memberikan peningkatan jaminan keamanan. Penulis memilih RSA-PSS untuk menguji kinerja RSA.

### C. ECDSA

ECDSA (Elliptic Curve Digital Signature Algorithm) adalah skema tanda tangan digital yang aman secara kriptografis yang berdasarkan kriptografi kurva eliptik (ECC). ECDSA bergantung pada matematika kelompok siklik kurva eliptik di atas bidang terbatas dan pada kesulitan masalah ECDLP (masalah logaritma diskrit kurva eliptik). Algoritma tanda / verifikasi ECDSA bergantung pada perkalian titik atau koordinat (x,y) EC. Kunci dan tanda tangan ECDSA lebih pendek daripada milik RSA untuk tingkat keamanan yang sama. Tanda tangan ECDSA 256-bit memiliki kekuatan keamanan yang sama seperti tanda tangan RSA 3072-bit. Adapun proses tanda tangan dan verifikasi ECDSA {r, s} dijelaskan sebagai berikut (Svetlin Nakov, 2018).

Langkah-langkah utama ECDSA adalah sebagai berikut.

1. Pembuatan tanda tangan

• Awalnya, parameter kurva (q,FR,a,b,G,n,h) harus ditetapkan pada tingkat keamanan yang diinginkan.

• Menghasilkan pasangan kunci yang cocok untuk kriptografi kurva eliptik, terdiri dari kunci privat dA (bilangan bulat yang dipilih secara acak dalam interval [1,n 1] ) dan kunci publik QA ( di mana QA = dAG ).

• Menghitung e = HASH(m), di mana HASH adalah fungsi hash kriptografik, seperti SHA-1.

• Pilih bilangan bulat acak k dari [1,n 1].

• Menghitung r = x1(mod n), dimana (x1,y1) = kG. Jika r = 0, pilih bilangan bulat acak yang berbeda k dari [1,n 1] lagi.

• Menghitung $s = k^{-1} (e + rd_A) (\text{mod } n)$. Jika s = 0, pilih bilangan bulat acak yang berbeda k dari [1,n 1] lagi.

• Tanda tangan adalah pasangan (r,s).

2. Verifikasi tanda tangan

• Menerima tanda tangan (r,s) dan QA kunci publik dari pengirim.

• Pastikan r dan s adalah bilangan bulat dalam [1,n 1]. Jika tidak, tanda tangan tidak valid.

• Menghitung e = HASH(m), di mana HASH adalah fungsi yang sama yang digunakan dalam pembuatan tanda tangan.

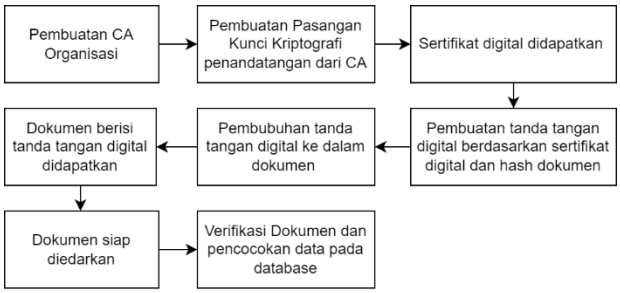
• Menghitung w $= (S)^{-1}$ (mod n).

• Menghitung u1 = ew(mod n) dan u2 = rw(mod n). Tanda tangan valid jika r = x1(mod n), tidak valid jika tidak.

## 4. PENGUJIAN

Penelitian ini menggunakan metodologi eksperimental yang bertujuan untuk mengukur performa masing-masing algoritma saat melakukan proses penandatanganan dokumen digital pada aplikasi Digisign. Spesifikasi perangkat keras yang digunakan adalah Server virtual CPU 2vCore, 1 GB RAM, 10 GB Storage. Sistem operasi Ubuntu server v21.04 LTS. Sementara itu *Librarry* yang digunakan dalam penelitian ini antara lain PyHanko (Python *based*) untuk pembubuhan tanda tangan ke PDF, Cryptography (Python), dan OpenSSL untuk pembuatan pasangan kunci tiap algoritma.

Alur dalam sistem yang dibangun dapat ditunjukkan pada gambar 4.1 berikut




Gambar 4. 1 Alur Sistem

Aplikasi Digisign UTD PNJ adalah media pengesahan dan verifikasi dokumen digital berbasis web, yang menerapkan teknologi tanda tangan digital. Algoritma tanda tangan digital yang akan diujikan berdasarkan *digital signature standard* yang diterbitkan oleh organisasi NIST, yaitu algoritma tanda tangan digital DSA, RSA, dan ECDSA. Panjang kunci yang digunakan DSA/RSA/ECDSA pada penelitian ini adalah 2048/2048/256 bit dengan mempertimbangkan waktu penggunaan dan keamanan (NIST, 2015), sebab panjang kunci ECDSA memiliki tingkat keamanan yang setara dengan panjang kunci RSA dan DSA yang lebih panjang. Selain itu, semakin panjang kunci yang digunakan akan mempengaruhi proses tanda tangan secara keseluruhan, termasuk proses verifikasi.

Dalam penelitian ini parameter yang diperhatikan adalah penggunaan waktu, penggunaan CPU, dan penggunaan *Memory* dalam 3 proses:

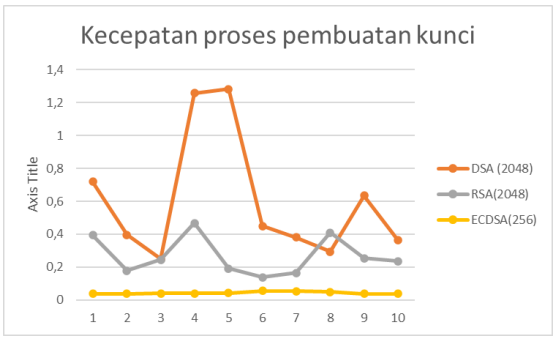
1. Pembuatan pasangan kunci,
2. Penandatanganan dokumen, dan
3. Verifikasi dokumen;

Pada proses pembuatan pasangan kunci diukur pada skema tersendiri yaitu pembuatan pasangan kunci dari masing-masing algoritma. Berikut perbandingan waktu yang dibutuhkan masing-masing algoritma dalam

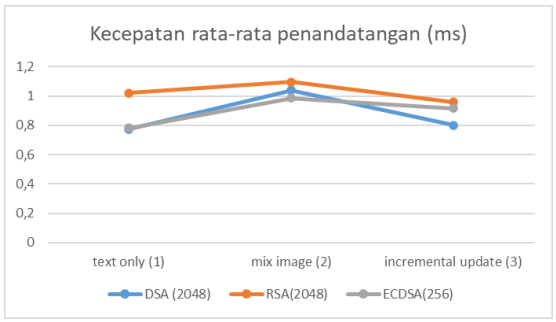
membuat pasangan kunci asimetris dalam 10 kali percobaan.

Pada proses penandatanganan dokumen dan verifikasi dokumen juga diukur dan diambil parameter penggunaan waktu. Masing-masing algoritma akan melakukan proses penandatanganan dokumen dan verifikasi dengan 3 skema dokumen yang berisi hanya teks (1) (35,6 KB), campuran gambar (2) (15 MB), dan melakukan tanda tangan pada dokumen yang telah ditandatangani (3) (35-37 KB bergantung pada algoritma yang digunakan).

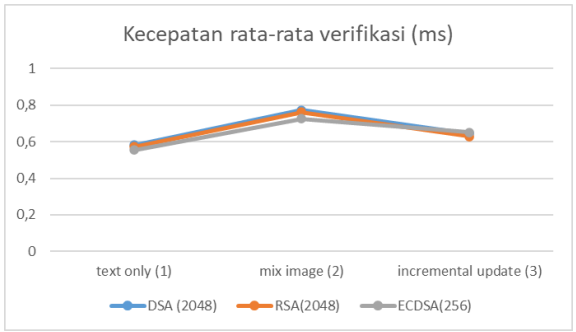
### 4.1. Analisis Pengukuran waktu




Gambar 4. 2 Grafik rata-rata waktu pada proses pembuatan pasangan kunci.



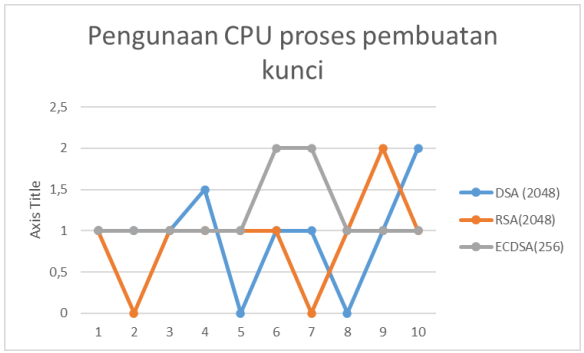

Gambar 4. 3 Grafik rata-rata waktu pada proses penandatanganan dokumen



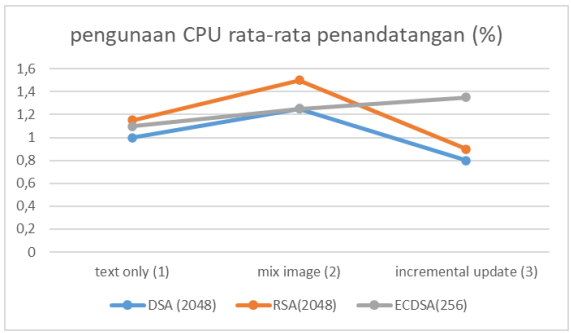

Gambar 4. 4 Grafik rata2 waktu pada proses verifikasi dokumen

Nilai penggunaan waktu yang semakin rendah menunjukkan semakin cepat proses selesai dilakukan, yang berarti semakin cepat suatu algoritma dalam melakukan suatu proses semakin baik algoritma tersebut. Secara keseluruhan berdasarkan Gambar 4.1-4 algoritma ECDSA memiliki performa terbaik dalam penggunaan waktu. Dari 3 proses dan 3 skema, algoritma ini hampir selalu menduduki peringkat pertama.

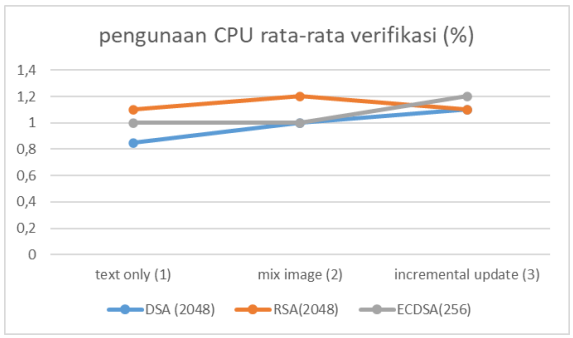
### 4.2. Analisis Pengukuran CPU




Gambar 4. 5 Grafik rata2 penggunaan CPU pada proses pembuatan pasangan kunci



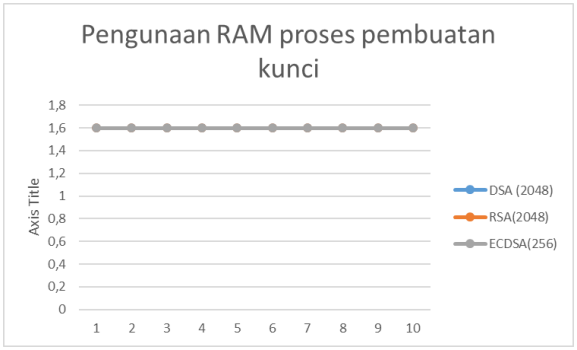

Gambar 4. 6 Grafik rata2 penggunaan CPU pada proses penandatanganan dokumen



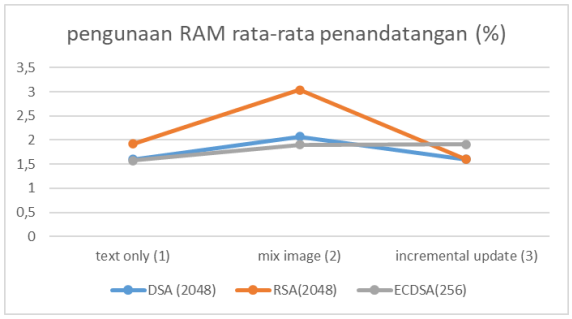

Gambar 4. 7 Grafik rata2 penggunaan CPU pada proses verifikasi dokumen

Nilai penggunaan CPU yang semakin rendah menunjukkan semakin cepat proses selesai dilakukan dan sebaliknya jika semakin tinggi penggunaan CPU yang dibutuhkan menunjukkan semakin kompleks algoritma tersebut. Secara keseluruhan berdasarkan Gambar 4.5-7 algoritma DSA memiliki performa terbaik dalam penggunaan CPU. Dari 3 proses dan 3 skema, algoritma ini hampir selalu menduduki peringkat pertama. Pada posisi kedua terdapat algoritma ECDSA yang terbilang cukup baik dan memiliki selisih yang tidak jauh dari DSA.

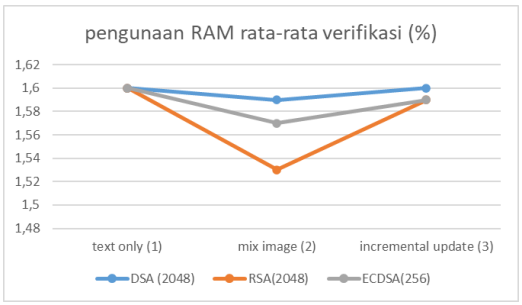
### 4.3. Analisis Pengukuran Memory




Gambar 4. 8 Grafik rata2 penggunaan *memory* pada proses pembuatan pasangan kunci




Gambar 4. 9 Grafik rata2 penggunaan *memory* pada proses penandatanganan dokumen




Gambar 4. 10 Grafik rata2 penggunaan *memory* pada proses verifikasi dokumen

Penggunaan memory yang semakin rendah memiliki nilai yang lebih baik dibandingkan dengan penggunaan memory yang lebih tinggi. Dengan kata lain, proses operasi pembuatan kunci, penandatanganan dokumen, dan verifikasi dokumen tidak menjadi beban pada server. Secara keseluruhan berdasarkan Gambar 4.8-10 algoritma RSA memiliki performa terbaik dalam penggunaan memory. Dari 3 proses dan 3 skema, algoritma ini beberapa kali menduduki peringkat pertama.

## 5. KESIMPULAN

Berdasarkan hasil dari penelitian yang telah dilakukan, dapat ditarik kesimpulan bahwa:

1. Performa algoritma tandatangan digital pada aplikasi Digisign dilakukan dengan 3 skema yang berbeda dan dalam 3 proses yang berbeda. Algoritma ECDSA unggul dalam penggunaan waktu pada proses pembuatan pasangan kunci, proses penandatanganan dokumen, dan proses verifikasi dokumen. Sementara itu algoritma DSA unggul dalam parameter penggunaan CPU dari ketiga proses. Disisi lain algoritma RSA lebih unggul dalam penggunaan memory pada ketiga proses.
2. Parameter yang penggunaan waktu, penggunaan CPU, dan penggunaan memory mampu menjadi parameter tolak ukur untuk menentukan algoritma tanda tangan digital yang paling sesuai untuk aplikasi Digisign.

## DAFTAR PUSTAKA

Afrianto, I. *et al.* (2020) 'Prototype of E-Document Application Based on Digital Signatures to Support Digital Document Authentication', *IOP Conference Series: Materials Science and Engineering*, 879(1). doi:10.1088/1757-899X/879/1/012042.

Arisandi, D., Sukri and Yusuf, M.B. (2020) 'Pemeriksaan Integritas Dokumen Dengan Digital Signature Algorithm', 4(1), pp. 1–6.

Cote, C. (2021) *What is Data Integrity and Why Does it Matter?*, *Harvard Business School Online*. Available at: https://online.hbs.edu/blog/post/what-is-data-integrity.

Dr. Abdel-Rahman Ismail (2014) 'Research Methodologies in Information Technology Research: A Comparative Study', *Implementation Science*, 39(1), pp. 1–15. Available at: http://dx.doi.org/10.1016/j.biochi.2015.03.025%0A

Finandhita, A. and Afrianto, I. (2018) 'Development of E-Diploma System Model with Digital Signature Authentication', *IOP Conference Series: Materials Science and Engineering*, 407(1). doi:10.1088/1757-899X/407/1/012109.

Fox, P. (2022) *Central Processing Unit (CPU), Khan Academy*. Available at: https://www.khanacademy.org/computing/computers-and-internet/xcae6f4a7ff015e7d:computers/xcae6f4a7ff015e7d:computer-components/a/central-processing-unit-cpu (Accessed: 14 July 2022).

IBM (2021) 'Why is data security important?' Available at: https://www.ibm.com/topics/data-security.

IONOS (2020) *High CPU usage*. Available at: https://www.ionos.com/digitalguide/server/know-

how/cpu-usage/ (Accessed: 14 July 2022).

Kaspersky (2021) *What is Data Encryption ?*, *Kaspersky*. Available at: https://www.kaspersky.com/resource-center/definitions/encryption.

Kavin, B.P. and Ganapathy, S. (2021) 'A new digital signature algorithm for ensuring the data integrity in cloud using elliptic curves', *International Arab Journal of Information Technology*, 18(2), pp. 180–190. doi:10.34028/IAJIT/18/2/6.

Muhtadin, P. (2017) 'Implementasi Digital Signature Dalam Validasi Online Local E-Government Menggunakan Algoritme Rsa Dan Md5 Prasta Muhtadin'.

NIST (2015) 'Digital Signature Standard', *Safeguarding Critical E-Documents*, (July), pp. 221–221. doi:10.1002/9781119204909.app1.

Nugraha, P. (2017) 'Implementasi Digital Signature Pada File Text Dengan Menggunakan Algoritma Schnorr Berbasis Android'.

Oyinola, J.M. (2020) *Authentication In A Body Area Network (Ban) using Openssl*. Fredericton.

Panda Security (2020) *What is RAM*. Available at: https://www.pandasecurity.com/en/mediacenter/tips/how-to-free-up-ram/ (Accessed: 14 July 2022).

Prabowo, E.C. and Afrianto, I. (2017) 'Penerapan Digital Signature Dan Kriptografi Pada Otentikasi Sertifikat Tanah Digital', *Komputa : Jurnal Ilmiah Komputer dan Informatika*, 6(2), pp. 83–90. doi:10.34010/komputa.v6i2.2481.

Pramono, P.P. (2019) 'Pendeteksian Dini Tingkat Kemanan Informasi Berbasis Iso 27001 : 2013 Menggunakan Metode Ahp (Analytical Hierarchy Process)', *Cyber Security dan Forensik Digital*, 2(2), pp. 57–64. doi:10.14421/csecurity.2019.2.2.1480.

Ramesh, A. and Suruliandi, A. (2013) 'Performance analysis of encryption algorithms for information security', *Proceedings of IEEE International Conference on Circuit, Power and Computing Technologies, ICCPCT 2013*, pp. 840–844. doi:10.1109/ICCPCT.2013.6528957.

Saepulrohman, A. and Ismangil, A. (2021) 'Data integrity and security of digital signatures on electronic systems using the digital signature algorithm (DSA) Agus Ismangil', *International Journal of Electronics and Communications System*, 1(1), pp. 11–15. Available at: http://ejournal.radenintan.ac.id/index.php/IJECS/index://creativecommons.org/licenses/by-sa/4.0/.

Somad, W.A. (2013) 'Sistem tanda tangan digital Online Untuk Naskah Dinas Menggunakan Algoritma Dsa (Digital Signature Algorithm)'.

Sukumaran, J. (2020) *Syrupy System Resource Usage Profiler*. Available at: https://github.com/jeetsukumaran/Syrupy.

Svetlin Nakov, P. (2018) 'Practical Cryptography for Developers', in *SoftUni (Software University)*. Sofia, Bulagria: MIT License. Available at: https://cryptobook.nakov.com/digital-signatures (Accessed: 29 March 2022).

The openssl project (2021) *OpenSSL*. Available at: https://www.openssl.org/ (Accessed: 11 May 2022).

Toradmalle, D. *et al.* (2018) 'Prominence Of ECDSA Over RSA Digital Signature Algorithm', *2018 2nd International Conference on I-SMAC (IoT in Social, Mobile, Analytics and Cloud) (I-SMAC)I-SMAC (IoT in Social, Mobile, Analytics and Cloud) (I-SMAC), 2018 2nd International Conference on*, pp. 253–257. doi:10.1109/I-SMAC.2018.8653689.

Valvekens, M. (2022) *PyHanko*. Available at: https://github.com/MatthiasValvekens/pyHanko (Accessed: 11 May 2022).