IMPLEMENTASI ALGORITME GRAIN SEBAGAI PROSES ENKRIPSI DATA DAN ALGORITME SHA-3 SEBAGAI *HASHING* DATA PADA RANCANG BANGUN APLIKASI PEMILU *ONLINE* BERBASIS *ANDROID*

**SKRIPSI**

Untuk memenuhi sebagian persyaratan

memperoleh gelar Sarjana Komputer

Disusun oleh:

Arsana Yudistira

NIM: 145150207111051



PROGRAM STUDI TEKNIK INFORMATIKA

JURUSAN TEKNIK INFORMATIKA

FAKULTAS ILMU KOMPUTER

UNIVERSITAS BRAWIJAYA  
MALANG

PENGESAHAN

IMPLEMENTASI ALGORITME GRAIN SEBAGAI PROSES ENKRIPSI DATA DAN ALGORITME SHA-3 SEBAGAI *HASHING* DATA PADA RANCANG BANGUN APLIKASI PEMILU *ONLINE* BERBASIS *ANDROID*

SKRIPSI

Diajukan untuk memenuhi sebagian persyaratan

memperoleh gelar Sarjana Komputer

Disusun Oleh :

Arsana Yudistira

NIM: 145150207111051

Skripsi ini telah diuji dan dinyatakan lulus pada

Telah diperiksa dan disetujui oleh:

|  |  |
| --- | --- |
| Dosen Pembimbing I  Ari Kusyanti, S.T, M.Sc  NIK: 201102 831228 2 001 | Dosen Pembimbing II  Lutfi Fanani, S.Kom., M.T., M.Sc.  NIK: 201607 890217 1 000 |

Mengetahui

Ketua Jurusan Teknik Informatika

Tri Astoto Kurniawan, S.T, M.T, Ph.D

NIP: 19710518 200312 1 001

PERNYATAAN ORISINALITAS

Saya menyatakan dengan sebenar-benarnya bahwa sepanjang pengetahuan saya, di dalam naskah skripsi ini tidak terdapat karya ilmiah yang pernah diajukan oleh orang lain untuk memperoleh gelar akademik di suatu perguruan tinggi, dan tidak terdapat karya atau pendapat yang pernah ditulis atau diterbitkan oleh orang lain, kecuali yang secara tertulis disitasi dalam naskah ini dan disebutkan dalam daftar pustaka.

Apabila ternyata didalam naskah skripsi ini dapat dibuktikan terdapat unsur-unsur plagiasi, saya bersedia skripsi ini digugurkan dan gelar akademik yang telah saya peroleh (sarjana) dibatalkan, serta diproses sesuai dengan peraturan perundang-undangan yang berlaku (UU No. 20 Tahun 2003, Pasal 25 ayat 2 dan Pasal 70).

Malang, 22 Januari 2018

­

Arsana Yudistira

NIM: 145150207111051

KATA PENGANTAR

Assalamualaikum Wr. Wb. Alhamdulillah, puji syukur penulis panjatkan kehadirat Allah SWT atas limpahan rahmat dan hidayahnya sehingga skripsi ini dapat terselesaikan dengan baik. Skripsi ini disusun sebagai salah satu syarat untuk memperoleh gelar Sarjana Komputer dari jurusan Teknik Informatika Fakultas Ilmu Komputer Universitas Brawijaya.

Penulis menyadari bahwa tanpa bantuan, bimbingan, serta dorongan dari semua pihak, penyelesaian skripsi ini tidak mungkin bisa terwujud. Pada kesempatan ini penulis menyampaikan rasa terima kasih sebesar-besarnya kepada:

1. Ibu Era Tresna Widya dan Bapak Nurhadi yang memberikan kasih sayang, dukungan dan doa yang tiada henti.
2. Bapak Wayan Firdaus Mahmudy, S.Si, M.T, Ph.D selaku Dekan Fakultas Ilmu Komputer Universtitas Brawijaya.
3. Bapak Tri Astoto Kurniawan , S.T, M.T, Ph.D selaku Ketua Jurusan Teknik Informatika Fakultas Ilmu Komputer Universitas Brawijaya.
4. Bapak Agus Wahyu Widodo , S.T, M.Cs selaku Ketua Program Studi Teknik Informatika Fakultas Ilmu Komputer Universitas Brawijaya .
5. Ibu Ari Kusyanti, S.T, M.Sc selaku dosen pembimbing pertama yang telah memberikan bimbingan, saran, motivasi, dan pengarahan dalam penyusunan skripsi ini.
6. Bapak Lutfi Fanani, S.Kom., M.T., M.Sc selaku dosen pembimbing kedua yang telah memberikan bimbingan, saran, motivasi, dan pengarahan dalam penyusunan skripsi ini.
7. Bapak, Ibu dosen serta segenap staff dan karyawan Jurusan Teknik Informatika yang telah membantu dalam menyelesaikan skripsi ini.
8. Ayudha Permata Putri dan Andira Faqih Muhammad, terima kasih untuk waktu, semangat, dukungan dan segala bentuk bantuan yang luar biasa.
9. Sahabat terbaik : Aditya Sandiva , Okky Anggada, Eka Pausya, dan Vito Bhaskara terima kasih untuk semua waktu dan nasihat-nasihatnya.
10. Teman-teman BE’M FILKOM 2014, Teman Beragam dan Kos Papa Ungu yang istimewa terima kasih untuk semua waktu dan momen-momen berharganya .
11. Teman-teman seperjuangan Teknik Informatika 2014 yang keren dan luar biasa, terima kasih untuk hal-hal yang luar biasa, tetap semangat dan semoga kita bisa lebih baik lagi ke depannya.
12. Teman-Teman seperjuangan skripsi Andre Rizal Sinaga , Kevin Charlie , Salma Mutia , Bagus Satria Wiguna semoga selalu diberi kelancaran oleh Allah SWT dalam penyelesaian skripsi dan semoga kita meraih kesuksesan setelah menyelesaikan skripsi ini .
13. Semua pihak yang telah memberikan bantuan baik secara langsung maupun tidak langsung dalam penyusunan skripsi ini.

Dalam penyusunan skripsi ini penulis menyadari bahwa skripsi ini belum sempurna karena keterbatasan ilmu dan kendala kendala lain yang terjadi selama pengerjaan skripsi ini. Semoga tulisan ini dapat bermanfaat dan dapat digunakan untuk pengembangan lebih lanjut. Wassalamualaikum Wr. Wb.

Malang,8 Januari 2018

Penulis

arsanay@gmail.com

ABSTRAK

Pemilihan umum atau biasa disebut pemilu sudah menjadi agenda rutin yang dilaksanakan oleh masyarakat Indonesia . Pada tahun 2014 tercatat pemerintah telah mengeluarkan biaya sebesar 24,1 triliun untuk menjalankan pemilu pada tahun tersebut . Disisi lain peningkatan teknologi tidak dapat dihalangi , salah satu buktinya adalah peningkatan jumlah penjualan *smartphone* berbasis *android* yang dicatat oleh web Statista.com . Pada tahun 2014 penjualan *smartphone* *android* menyentuh angka 250juta unit. Oleh karena itu tingginya jumlah penjualan *smartphone* dapat dimanfaatkan sebagai solusi dari besarnya anggaran yang dikeluarkan pemerintah untuk melaksanakan kegiatan pemilu dengan cara merubah sistem pemilu konvensional menjadi pemilu berbasis *online* atau E-Votting . Dalam mewujudkan kegiatan E-Votting aspek keamanan data perlu diperhatikan karena pemilu memiliki sifat rahasia . Menerapkan algoritme grain untuk mengenkripsi data dan algoritme SHA-3 untuk melakukan *hashing* pada data cukup untuk memberikan asas “Rahasia” pada peserta pemilu . Pada penelitian ini penulis melakukan enkripsi data pada data pasangan calon yang dipilih oleh peserta pemilu lalu hasil enkripsi berupa *cipher text* akan di hash dan menghasilkan *output* berupa *digest text* , dimana *digest text* tersebut yang akan disimpan pada database. Terdapat dua jenis pengujian yang dilakukan untuk mengetahui apakah algoritme yang digunakan pada penelitian ini cocok diterapkan pada aplikasi pemilu *online* , yang pertama yaitu pengujian waktu pemrosesan dan yang kedua ialah pengujian *avalanche effect* . Pada pengujian waktu pemrosesan dilakukan 30 kali percobaan. Skenario yang diterapkan ialah algoritme grain berjalan diikuti dengan SHA-3 . Dari keseluruhan percobaan didapatkan rata-rata nilai tempuh waktu pemrosesan yang relatif sangat kecil yaitu 7.2 ms ketika dijalankan pada *Netbeans ide*, namun ketika di lakukan pengujian pada *device* *android* nilai waktu pemrosesan meningkat menjadi 153 ms . Pada pengujian *avalanche* *effect*, algoritme grain mendapatkan nilai sebesar 44.6 % , algoritme SHA-3 mendapatkan 98.2 % sedangkan ketika algoritme grain berjalan dilanjutkan dengan SHA-3 akan mendapatkan nilai *avalanche* *effect* sebesar 96.4 %. Hasil tersebut membuktikan bahwa melakukan kombinasi algoritme grain yang dilanjutkan dengan SHA-3 akan meningkatkan nilai *avalanche* *effect* .Persentase minimum *avalanche effect* untuk sebuah algoritme agar dapat dikatakan baik adalah lebih dari 50%. Kesimpulan dari pengujian diatas ialah kombinasi algoritme grain dan SHA-3 sangat baik dan cocok jika diterapkan pada aplikasi pemilu *online* berbasis *android* karena memiliki waktu pemrosesan yang rendah dan juga memiliki nilai *avalanche* *effect* yang tinggi.

**Kata Kunci:** aplikasi *android*,*e-votting*, enkripsi, fungsi *hash*, kriptografi, algoritme Grain, algoritme SHA-3.

*ABSTRACT*

*General election has become a routinity for Indonesian people. In 2014 the government has been pay out 24.1 trillion to hold the election in that year. On the other hand the increasing in technology can not be avoided, one of evidance is the increasing in the number of Android-based smartphone sales according to statista.com. In 2014 Android based smartphone reached 250 million units. Therefore, high number of smartphones sellings can be used as a solution from the amount of budget which spent by the government to conduct election activities by changing conventional election system to online-based election or E-vote. In realizing the election activities, aspects of data security need to be noticed because election has confidentiality. Applying the algorithm grain to encrypt datas and algorithm SHA-3 to hash the datas consider sufficient to provide secret principle for the election participants . In this research , the writer do the encryption data of the election candidate that chosen by participants then the output of encryption will be hash to gIVe an output as digest text , the digest text will be stored at database. There are two types of testing to find out whether the algorithm that used in this research match to applied on e-vote application , the first one is time processing testing and the second one is avalanche effect testing . On time processing testing , the testing done 30 times . Scenario that applied was grain algorithm run first and following by SHA-3 . The entire experiments was obtained a result that said if the scenario run on Netbeans ide need 7.2 ms to processing the input to get an output , however if the scenario run on device Android the time processing will increase to 153 ms . On avalanche effect testing , grain algorithm get 44.6 % on percentage of change , while SHA-3 get 98.2 % . If grain runs first and following by SHA-3 after that , the result of avalanche effect is 96.4% . Those result proof that combining those algorithm will increase the result of avalanche effect testing . Precentage minimum score of avalanche effect testing for every algorithm to proof the algorithm run well is more than 50% . The conclusion of those testing is combination of grain and SHA-3 run very well and match if applied on e-vote application that runs on Android platform because has low time processing and also has high avalanche effect score .*

***Keywords:*** *android app, e-votting, encryption, hash function, cryptography, Grain algorithm, SHA-3 algorithm.*

# DAFTAR ISI

[PENGESAHAN i](#_Toc503157899)

[PERNYATAAN ORISINALITAS ii](#_Toc503157900)

[KATA PENGANTAR iii](#_Toc503157901)

[ABSTRAK v](#_Toc503157902)

[*ABSTRACT* vi](#_Toc503157903)

[DAFTAR ISI vii](#_Toc503157904)

[DAFTAR TABEL x](#_Toc503157905)

[DAFTAR GAMBAR xi](#_Toc503157906)

[DAFTAR CODE xii](#_Toc503157907)

[BAB 1 PENDAHULUAN 1](#_Toc503157908)

[1.1 Latar Belakang 1](#_Toc503157909)

[1.2 Rumusan Masalah 2](#_Toc503157910)

[1.3 Tujuan 2](#_Toc503157911)

[1.4 Manfaat 2](#_Toc503157912)

[1.5 Batasan masalah 3](#_Toc503157913)

[1.6 Sistematika Penulisan 3](#_Toc503157914)

[BAB 2 KAJIAN PUSTAKA DAN DASAR TEORI 5](#_Toc503157915)

[2.1 Kajian Pustaka 5](#_Toc503157916)

[2.2 Dasar Teori 5](#_Toc503157917)

[2.2.1 Pemilihan Umum 6](#_Toc503157918)

[2.2.2 Pemilu *Online* (E-Voting) 6](#_Toc503157919)

[2.3 Sistem dan Sistem Keamanan 6](#_Toc503157920)

[2.4 Kriptografi 7](#_Toc503157921)

[2.5 Algoritme Grain 9](#_Toc503157922)

[2.5.1 Inisialisasi Kunci 11](#_Toc503157923)

[2.6 Algoritme SHA3 12](#_Toc503157924)

[2.7 Perangkat Bergerak 14](#_Toc503157925)

[2.7.1 Aplikasi Perangkat Bergerak 15](#_Toc503157926)

[2.8 *Client* *Server* 15](#_Toc503157927)

[2.8.1 Sistem *Client Server* 15](#_Toc503157928)

[BAB 3 METODOLOGI 17](#_Toc503157929)

[3.1 Studi Literatur 18](#_Toc503157930)

[3.2 Analisis Kebutuhan 18](#_Toc503157931)

[3.3 Perancangan Sistem 19](#_Toc503157932)

[3.4 Implementasi 19](#_Toc503157933)

[3.5 Pengujian dan Analisis 19](#_Toc503157934)

[3.6 Pengambilan Kesimpulan dan Saran 20](#_Toc503157935)

[BAB 4 ANALISIS DAN PERANCANGAN 21](#_Toc503157936)

[4.1 Analisis Kebutuhan 21](#_Toc503157937)

[4.1.1 Analisis Permasalahan 21](#_Toc503157938)

[4.1.2 Analisis Data 24](#_Toc503157939)

[4.1.3 Analisis Keamanan Data 24](#_Toc503157940)

[4.2 Perancangan Sistem 25](#_Toc503157941)

[4.2.1 Flowchart Sistem 25](#_Toc503157942)

[4.3 Desain Tampilan 26](#_Toc503157943)

[BAB 5 IMPLEMENTASI DAN PENGUJIAN 29](#_Toc503157944)

[5.1 Implementasi 29](#_Toc503157945)

[5.1.1 Algoritme Grain 30](#_Toc503157946)

[5.1.2 Implementasi Pada Proses Pemilihan 35](#_Toc503157947)

[5.2 Parameter Pengujian 36](#_Toc503157948)

[5.3 *Test Vector* 36](#_Toc503157949)

[5.3.1 Tujuan Pengujian 36](#_Toc503157950)

[5.3.2 Prosedur Pengujian 36](#_Toc503157951)

[5.3.3 Hasil dan Analisis 36](#_Toc503157952)

[5.4 Pengujian Pemrosesan *Cipher text* 38](#_Toc503157953)

[5.4.1 Tujuan Pengujian 38](#_Toc503157954)

[5.4.2 Prosedur Pengujian 38](#_Toc503157955)

[5.4.3 Hasil Pengujian 38](#_Toc503157956)

[5.5 Pengujian Pemrosesan *Digest text* 41](#_Toc503157957)

[5.5.1 Tujuan Pengujian 41](#_Toc503157958)

[5.5.2 Prosedur Pengujian 41](#_Toc503157959)

[5.5.3 Hasil Pengujian 42](#_Toc503157960)

[5.6 Pengujian Pemrosesan Seluruh Sistem 45](#_Toc503157961)

[5.6.1 Tujuan Pengujian 45](#_Toc503157962)

[5.6.2 Prosedur Pengujian 45](#_Toc503157963)

[5.6.3 Hasil Pengujian 45](#_Toc503157964)

[5.7 Pengujian *Confidentiality* 48](#_Toc503157965)

[5.7.1 Tujuan Pengujian 48](#_Toc503157966)

[5.7.2 Prosedur Pengujian 48](#_Toc503157967)

[5.7.3 Hasil Pengujian 48](#_Toc503157968)

[5.8 Pengujian *Integrity* 50](#_Toc503157969)

[5.8.1 Tujuan Pengujian 50](#_Toc503157970)

[5.8.2 Prosedur Pengujian 51](#_Toc503157971)

[5.8.3 Hasil Pengujian 51](#_Toc503157972)

[5.9 Pengujian Skema Pembentukan Key 53](#_Toc503157973)

[5.9.1 Tujuan Pengujian 53](#_Toc503157974)

[5.9.2 Prosedur Pengujian 53](#_Toc503157975)

[5.9.3 Hasil Pengujian 53](#_Toc503157976)

[5.10 Pengujian *Avalanche Effect* 54](#_Toc503157977)

[5.10.1 Tujuan pengujian 54](#_Toc503157978)

[5.10.2 Prosedur pengujian 55](#_Toc503157979)

[5.10.3 Hasil pengujian 55](#_Toc503157980)

[5.10.4 Analisis pengujian 56](#_Toc503157981)

[BAB 6 PENUTUP 57](#_Toc503157982)

[6.1 Kesimpulan 57](#_Toc503157983)

[6.2 Saran 58](#_Toc503157984)

[Daftar Pustaka 59](#_Toc503157985)

# DAFTAR TABEL

[Tabel 5.1 Tabel Spesifikasi Perangkat Lunak dan Perangkat Keras 29](#_Toc504219195)

[Tabel 5. 2 *Source Code* Inisialisasi LFSR 30](#_Toc504219196)

[Tabel 5. 3 *Source Code* Aritmatika LFSR 31](#_Toc504219197)

[Tabel 5. 4 *Source Code* Inisialisasi NFSR 31](#_Toc504219198)

[Tabel 5. 5 *Source Code* Aritmatika NFSR 32](#_Toc504219199)

[Tabel 5. 6 *Source Code* Filter *Function* 32](#_Toc504219200)

[Tabel 5. 7 *Source Code* Z *Function* 33](#_Toc504219201)

[Tabel 5. 8 *Source Code* Inisialisasi Kunci 33](#_Toc504219202)

[Tabel 5. 9 *Source Code* Pembuatan Nilai *Keystream* 34](#_Toc504219203)

[Tabel 5. 10 *Source Code* Fungsi Enkripsi 35](#_Toc504219204)

[Tabel 5. 11 *Source Code* Fungsi Dekripsi 35](#_Toc504219205)

[Tabel 5. 12 *Source Code* Pemanggilan Fungsi grain dan SHA-3 36](#_Toc504219206)

[Tabel 5. 13 Hasil pengujian waktu tempuh aplikasi pemilu online berbasis android dengan grain maupun tanpa grain 39](#_Toc504219207)

[Tabel 5. 14 *Independent samples t-Test* untuk pengujian waktu tempuh aplikasi pemilu online berbasis android dengan grain maupun tanpa grain 41](#_Toc504219208)

[Tabel 5. 15 Hasil pengujian waktu tempuh aplikasi pemilu online berbasis android dengan SHA-3 maupun tanpa SHA-3 43](#_Toc504219209)

[Tabel 5. 16 *Independent samples t-Test* untuk pengujian waktu tempuh aplikasi pemilu *online* berbasis android dengan SHA-3 maupun tanpa SHA-3 45](#_Toc504219210)

[Tabel 5. 17 Hasil pengujian waktu pemrosesan seluruh sistem 46](#_Toc504219211)

[Tabel 5. 18 *Independent samples t-Test* untuk pengujian waktu tempuh aplikasi pemilu online berbasis android dengan Grain dan SHA-3 maupun tanpa grain dan SHA-3 48](#_Toc504219212)

[Tabel 5. 19 Tabel pengujian *confidentiality* pertama 50](#_Toc504219213)

[Tabel 5. 20 Tabel pengujian *confidentiality* kedua 50](#_Toc504219214)

[Tabel 5. 21 Tabel pengujian *confidentiality* ketiga 50](#_Toc504219215)

[Tabel 5. 22 Tabel pengujian *confidentiality* keempat 51](#_Toc504219216)

[Tabel 5. 23 Tabel pengujian *confidentiality* kelima 51](#_Toc504219217)

[Tabel 5. 24 Tabel pengujian *integrity* pertama 52](#_Toc504219218)

[Tabel 5. 25 Tabel pengujian *integrity* kedua 52](#_Toc504219219)

[Tabel 5. 26 Tabel pengujian *integrity* ketiga 53](#_Toc504219220)

[Tabel 5. 27 Tabel pengujian *integrity* keempat 53](#_Toc504219221)

[Tabel 5. 28 Tabel pengujian *integrity* kelima 54](#_Toc504219222)

[Tabel 5. 29 Hasil Pengujian Skema Pembentukan Key 55](#_Toc504219223)

[Tabel 5. 30 Hasil pengujian *avalanche effect* untuk algoritme grain 56](#_Toc504219224)

[Tabel 5. 31 Hasil pengujian *avalanche effect* untuk algorimta SHA-3 56](#_Toc504219225)

[Tabel 5. 32 Hasil pengujian *avalanche effect* untuk algorimta grain dilanjutkan dengan SHA-3 57](#_Toc504219226)

# DAFTAR GAMBAR

[Gambar 2. 1 Skema Proses Enkriosi dan Dekripsi 8](#_Toc503142142)

[Gambar 2. 2 Diagram Umum Grain *Cipher* 11](#_Toc503142143)

[Gambar 2. 3 Inisialisasi Grain *Cipher* 12](#_Toc503142144)

[Gambar 4. 1 Skema Global Aplikasi Pemilu *Online*………………………………………………….21](#_Toc503142135)

[Gambar 4. 2 Skema Global Algoritme Grain 22](#_Toc503142136)

[Gambar 4. 3 Skema Global Algoritme SHA-3 23](file:///D:\DOKUMEN%20ASA\Dokumen%20Kuliah\Skripsi\Skripsi%20Arsana%20Yudistira%20BAB%201-6%20(Repaired).docx#_Toc503142137)

[Gambar 4. 4 Flowchart Sistem Aplikasi Pemilu *Online* 25](file:///D:\DOKUMEN%20ASA\Dokumen%20Kuliah\Skripsi\Skripsi%20Arsana%20Yudistira%20BAB%201-6%20(Repaired).docx#_Toc503142138)

[Gambar 4. 5 Desain Tampilan Halaman Login 26](file:///D:\DOKUMEN%20ASA\Dokumen%20Kuliah\Skripsi\Skripsi%20Arsana%20Yudistira%20BAB%201-6%20(Repaired).docx#_Toc503142139)

[Gambar 4. 6 Desain Tampilan Halaman Calon 27](file:///D:\DOKUMEN%20ASA\Dokumen%20Kuliah\Skripsi\Skripsi%20Arsana%20Yudistira%20BAB%201-6%20(Repaired).docx#_Toc503142140)

[Gambar 4. 7 Desain Tampilan Halaman Akhir 28](file:///D:\DOKUMEN%20ASA\Dokumen%20Kuliah\Skripsi\Skripsi%20Arsana%20Yudistira%20BAB%201-6%20(Repaired).docx#_Toc503142141)

[Gambar 5. 1 Grafik Hasil Waktu Tempuh Aplikasi Pemilu Online Berbasis Android dengan Grain maupun tanpa grain…………………………………………………………..…………….40](#_Toc503142132)

[Gambar 5. 2 Grafik Hasil Waktu Tempuh Aplikasi Pemilu *Online* Berbasis Android dengan SHA-3 maupun tanpa SHA-3 .43](#_Toc503142133)

[Gambar 5. 3 Grafik Hasil Waktu Tempuh Aplikasi Pemilu Online Berbasis Android dengan Grain dan SHA-3 maupun tanpa Grain dan SHA-3 47](#_Toc503142134)

# BAB 1 PENDAHULUAN

## 1.1 Latar Belakang

Pemilu adalah sarana yang tersedia bagi rakyat untuk menjalankan kedaulatannya sesuai dengan azas yang bermaktub dalam Pembukaan UUD 1945 oleh Ali Moertopo (Rainer, 2017). Sejak 1955, Indonesia telah menyelenggarakan 11 pemilu. Berdasarkan UUD 1945 pasal 1 ayat 3, memberikan hak kepada rakyat untuk menentukan kekuasaan.

Pada tahun 2014 tercatat pemerintah telah mengeluarkan biaya sebesar 24,1 triliun untuk menjalankan pemilu pada tahun tersebut (Maesaroh, 2014). Angka tersebut merupakan angka yang fantastis dan berpotensi dapat dipangkas agar APBN negara dapat digunakan lebih maksimal.

Disisi lain peningkatan teknologi tidak dapat dihalangi, salah satu buktinya adalah peningkatan jumlah penjualan *smartphone* berbasis *android* yang dicatat oleh web Statista.com. Tahun 2009 tercatat bahwa penjualan *smartphone* *android* jauh dibawah angka 50 juta unit namun pada tahun 2014 penjualan *smartphone* *android* menyentuh angka 250 juta unit. Umumnya masyarakat sudah memiliki handphone masing-masing, bahkan anak dibawah umur yang sudah menggunakan handphone bukan merupakan fenomena asing dewasa ini.

Dengan meningkatnya perkembangan teknologi diharapkan dapat menjadi celah guna mengurangi anggaran negara untuk melaksanakan pemilu secara langsung pada TPS. Pemilu berbasis *online* dapat menjadi trobosan baru yang sangat efektif menekan angka APBN untuk melaksanakan pemilu, dikarenakan banyak infrastruktur yang tidak perlu diadakan lagi sehingga dapat menekan tingginya APBN pelaksanaan pemilu. Kita dapat memanfaatkan Pemilu *online* dengan infrastruktur handphone masing-masing calon pemilih guna pengganti kertas.

Pemilu merupakan suatu kegiatan yang bersifat sangat rahasia dan *sensitive*. Data pilihan peserta pemilu tidak boleh diketahui siapapun termasuk pelaksana pemilu online dan juga pemegang otoritas database pada sistem pemilu online. Sehingga diperlukan enkripsi dan hash pada data yang dikirim dan disimpan pada database server. Hal ini juga diharapkan dapat menjaga tingkat *confidentiality* dan *integrity* peserta pemilu dengan menangkal aktifitas para penyadap data yang memiliki suatu kepentingan pribadi pada kegiatan pemilu.

Algoritme Grain merupakan algoritme enkripsi yang sampai saat ini belum ditemukan titik kelemahanya kecuali jika dibandingkan dengan efisiensi algoritme lain yang sengaja didesain untuk *software* dengan kecepatan tinggi. Algorime Grain dan SHA-3 tentu juga memiliki kelebihan yaitu Grain didesain untuk berjalan pada *hardware* dengan *resource* yang kecil, Grain merupakan algoritme yang berorientasi pada setiap bit bukan blok kata sehingga menurunkan nilai complexity dan meningkatkan kecepatan komputas (Hell Martin, Thomas Johanson dan Willi Meier, 2007). SHA-3 diresmikan pada tahun 2015 sehingga masih sangat sedikit penerapanya dan SHA-3 bekerja dengan cara melakukan hash terhadap sebuah dokumen, sehingga menghasilkan tanda tangan *digit*al untuk suatu dokumen (NIST ,2015).

Hal-hal tersebut lah yang menjadikan dasar penerapan Algoritme Grain dan SHA-3 pada Rancangan Pemilu *Online.*

## 1.2 Rumusan Masalah

Berdasarkan uraian latar belakang tersebut, maka dirumuskan masalah sebagai berikut :

1. Bagaimana waktu pemrosesan aplikasi pemilu online berbasis android ketika menerapkan algoritme Grain dan algoritme hash SHA-3 dibandingkan dengan tidak menerapkan algoritme Grain dan algoritme SHA-3?
2. Bagaimana algoritme grain dan SHA-3 dapat memberikan aspek *confidentiality* dan *integrity* terhadap pengguna aplikasi pemilu *online* ?
3. Berapa nilai *avalanche effect* algoritme grain, algoritme SHA-3 dan juga ketika kedua algoritme tersebut dikombinasikan ?

## 1.3 Tujuan

Tujuan dari penelitian ini adalah untuk menjawab rumusan masalah yang telah dijelaskan sebelumnya diatas, diantaranya adalah :

1. Mendapatkan hasil waktu pemrosesan aplikasi pemilu online berbasis android ketika menerapkan algoritme Grain dan algoritme hash SHA-3 dibandingkan dengan tidak menerapkan algoritme Grain dan algoritme SHA-3 .
2. Untuk mengetahui bagaimana algoritme grain dan SHA-3 dapat memberikan aspek *confidentiality* dan *integiry* terhadap pengguna aplikasi pemilu *online*
3. Mendapatkan nilai *avalanche effect* algoritme grain, algoritme SHA-3 dan juga ketika kedua algoritme tersebut dikombinasikan.

## 1.4 Manfaat

Manfaat yang didapatkan dari penelitian ini adalah :

1. Pengguna akan mendapatkan keamanan dan kearahasiaan informasi pada saat melakukan pemilihan pada pemilu *online* melalui teknik kriptografi dengan cara mengenkripsi data.
2. Pelaksana pemilu tidak lagi memerlukan anggaran untuk pengadaaan kertas sebagai media pemilihan calon.
3. Pengguna dapat menjaga keutuhan informasi (*integrity*) informasi pemilu untuk menghindari upaya penyadapan, pembajakan dan hal hal yang menyebabkan kebocoran dan manipulasi informasi dengan memanfaatkan teknik kirptografi guna mengenkripsi data.

## 1.5 Batasan masalah

Untuk menghindari adanya kemungkinan semakin berkembangnya masalah, maka penelitian dalam laporan ini dibatasi oleh hal-hal sebagai berikut :

1. Penelitian tidak membahas tampilan dan fungsional selain yang ditampilkan aplikasi ini pada saat pengujian.
2. Proses bisnis yang dilakukan mencakup enkripsi, hash dan dekripsi terhadap informasi pemilu *online.*
3. Pasangan calon yang ditampilkan dilengkapi tombol berbentuk *radio button* untuk melakukan proses pemilihan.
4. Algoritme SHA-3 pada penelitian ini menggunakan library eksternal.

## 1.6 Sistematika Penulisan

## Sistematika penulisan memberikan gambaran dan uraian dari penyusunan skripsi secara garis besar yang meliputi beberapa bab, antara lain :

**BAB I PENDAHULUAN**

Memuat latar belakang, rumusan masalah, batasan masalah, tujuan penelitian, manfaat penelitian, dan sistematika penulisan.

**BAB II TINJAUAN PUSATAKA**

Berisi kajian pusataka, referensi dan sumber-sumber yang berhubungan dengan permasalahan dalam skripsi antara lain mengenai pemilu *online*, sistem keamanan, kriptografi, algoritme Grain, SHA 3 serta teori-teori lainya sebagai dasar penulisan skripsi.

**BAB III METODOLOGI PENELITIAN**

Menjelaskan metode atau langkah-langkah yang digunakan dalam penulisan skripsi. Metode apa yang dugunakan baik dalam penulisan, perancangan, implementasi, dan pengujian dari *system* yang dibangun.

**BAB IV ANALISIS DAN PERANCANGAN**

Pada bab ini dijelaskan analisis dan perancangan *system* keamanan pemilu *online* dengan algoritme kriptografi memanfaatkan algoritme kriptografi Grain dan SHA3 yang dapat menjawab permasalahan yang telah diuraikan pada rumusan masalah.

**BAB V IMPLEMENTASI DAN PENGUJIAN**

Bab ini membahas tentang implementasi *system* keamanan dan pengujian berdasarkan metode penelitian yang telah dibuat untuk mengetahui hasilnya.

**BAB VI PENUTUP**

Bab ini memuat tentang kesimpulan yang diperoleh dari pembuatan dan pengujian aplikasi yang dikembangkan dalam penelitian skripsi disertai saran yang dapat dijadikan masukan untuk pengembangan lebih lanjut.

# 

# BAB 2 KAJIAN PUSTAKA DAN DASAR TEORI

Kajian pustaka dan dasar teori berutujuan untuk membentuk kerangka teori yang berasal dari *literature* yang berhubungan dengan masalah yang diteliti. Bab ini membahas tinjauan pustaka yang digunakan untuk menunjang penulisan skripsi, dimana dalam tinjauan pustaka terdapat kajian pustaka mengenai referensi dari penelitian sebelumnya yang terkait dengan kelebihan algoritme yang diimplementasikan dalam penelitian. Sedangkan dasar teori membahas mengenai teori-teori dasar yang berhubungan dengan konsep dasar pemilu, sistem keamanan, kriptografi, algoritme Grain, algoritme hash SHA3 dan perangkat bergerak.

## 2.1 Kajian Pustaka

Pada sub bab ini mmbahas tentang alasan mengapa skripsi ini menggunakan algoritme krptografi Grain dan hash SHA-3 berdasarkan jurnal penelitian terlebih dahulu. Penelitian sebelumnya berjudul “*Energy Eficiency of Encryption for Wireless Device*” (Elminaam, 2009) menyarankan untuk mempertimbangkan penggunaan *resource* batrei pada perangkat *wireless*, mengingat sangat terbatasnya sangat terbatasnya kapasistas batrei pada setiap perangkat. Daya batrei juga bergantung pada masalah energy yang dikonsumsi dikarenakan algoritme kriptografi yang digunakan. Dewasa ini telah adanya muncul suatu sebutan “*Battery Gap*” yang dikarenakan perkembangan baterai lebih lambat dibandingkan dengan perkembangan teknologi lainya, sehingga diperlukan cara untuk memanfaatkan *resource* atauapun kapasistas batrei dengan sebaik-baiknya dan seefisien mungkin. Pada penelitian lainya dijelaskan bahwa Algoritme Grain didesain untuk berjalan pada *hardware* dengan *resource* yang sedikit. Grain juga disebut dapat berjalan pada lingkungan dengan konsumsi power dan memori yang sangat terbatas. Namun Grain juga dapat ditingkatkan kecepatanya jika *resource* dari *hardware* memungkinkan (Hell Martin, Thomas Johanson dan Willi Meier, 2007). Tingkat integritas pada pemilu *online* juga sangat diperlukan sehingga penulis memutuskan untuk menambahkan fungsi hasing menggunakan SHA-3. Pada situs resmi *National Institute of Standards and Technology* (NIST) dikatakan bahwa *Keccak* merupakan pemenang dari kompetisi yang diadakan NIST dan juga dijadikan menjadikan *Keccak standart* untuk SHA-3 pada tahun 2015 menggantikan SHA-1.

## 2.2 Dasar Teori

Dasar teori membahas tentang materi-materi yang berkaitan dengan pengembangan aplikasi *system* keamanan aplikasi *client* pemilu *online* dengan algoritme kriptografi Grain dan teknik *hashing* SHA3. Hal-hal yang dibahas pada bagian ini adalah pemilu, *system* keamanan, kriptografi, algoritme Grain, algoritme hash SHA-3, platform *android*, aplikasi *client* *server* dan perangkat bergerak.

### 2.2.1 Pemilihan Umum

Pemilihan umum atau yang biasa lebih dikenal dengan pemilu adalah wujud nyata demokrasi prosedural, meskipun demokrasi tidak sama dengan pemilihan umum, namun pemilihan umum merupakan salah satu aspek demokrasi yang sangat penting yang juga harus diselenggarakan secara demokratis (Veri Junaidi, 2009).

Pemilu juga merupakan sarana perwujudan kedaulatan rakyat sekaligus merupakan arena kompetisi yang paling adil bagi partai politik, sejauh mana telah melaksanakan fungsi dan perannya serta pertanggung jawaban atas kinerjanya selama ini kepada rakyat yang telah memilihnya (Sukriono Didik, 2009).

### 2.2.2 Pemilu *Online* (E-Voting)

Definisi e-voting jika dicermati adalah tujuanya yaitu lebih mengacu kepada pemanfaatan perangkat elektronik untuk lebih memudahkan dan melancarkan proses dan mengotomatisasi segala kemungkianan camput tangan individu dalam tiap prosesnya (Smith dan Clark, 2005). Salah satu definisi *e-voting* adalah suatu *system* pemilihan dimana data dicatat, disimpan dan diproses dalam bentuk informasi *digital* (VoteHere Inc, 2002).

*E – voting* atau *electornic* *voting* secara sederhana dapat diartikan sebagai penggunaan hak pilih dalam sebuah peilu dengan menggunakan teknologi (secara elektronik) (Darmawan. et al., 2014)

### 2.3 Sistem dan Sistem Keamanan

Sistem bukan merupakan entitas yang berdiri sendiri, melainkan terdapat dalam suatu lingkugan. Lingkungan ini mempengaruhi fungsi dan kinerja *system*. Kadangkala lingkungan bisa dianggap sebagai *system* pula tetapi lebih umumnya, lingkungan terdiri dari sejumlah *system* lain yang berinteraksi satu dengan lainya (Sommerville Ian, 2003).

Sistem kemanan adalah tindakan pencegahan terhadap segala bentuk penyebab kerugian, termasuk didalamnya kerugian secara fisik dan non fisik, berwujud atau tidak berwujud, serta adanya bermacam-macam kerugian oleh berbagai sebab.

Garfinkel mengemukakan bahwa *system* keamanan melingkupi lima aspek, meliputi (Widiyanto, 2007) :

1. *Privacy* /*Confidentiality*

Aspek *privacy* atau *confidentiality* merupakan sebuah usaha untuk menjaga dan memberikan keamanan informasi seseorang agar tidak dapat diketahui oleh pihak lain. *Privacy* lebih kearah data-data yang bersifat rahasia sedangkan *confdentiality* berhubungan dengan data yang diberikan kepada pihak lain dengan maksud dan tujuan tertentu.

1. *Integrity*

Aspek *integrity* atau integritas merupakan usaha untuk menjaga dan menjamin bahwa suatu data tidak dapat dirubah-rubah tanpa seizin pemilik. Jika terdapat perbedaan maka boleh dibilang aspek integritas tidak tercapai.

1. *Authentication*

Aspek *authentication* adalah aspek yang menyataka bahwa sebuah informasi benar-benar asli, jika berhubungan dengan subjek berarti subjek tersebutlah yang benar-benar dimaksut.

1. *Availability*

Aspek *Availability* ialah jaminan bahwa data atau informasi yang dimaksut benar-benar ada atau tersedia.

1. *Access Control*

Aspek *Access Control* ialah aspek yang mengatur hak akses sebuah informasi agar sebuah informasi yang bersifat rahasia tidak dapat di akses orang yang tidak memiliki kepentingan. Misalnya, seorang *administrator* memiliki hak akses penuh terhadap sebuah *computer*, tetapi hal ini tidak berlaku bagi *account guest* ataupun *limited account* lainya.

## 2.4 Kriptografi

Kriptografi diambil dari Bahasa Yunani yaitu ”*crypthos*” dan “*graphein”*. *Crypto* berarti rahasia dan *graphein* berarti tulisan. Sehingga kriptografi memiliki arti yaitu tulisan rahasia. Kriptografi merupakan bidang ilmu tentang teknik enkripsi dimana data sengaja diacak menggunakan suatu algoritme enkripsi menjadi sesuatu yang sulit dibaca oleh seseorang yang tidak memiliki kunci dekripsi. Kegiatan dekripsi hanya dapat dilakukan dengan menggunakan kunci dekripsi untuk mendapatkan kembali data asli (Kromodimoeljo, 2009).

Kriptografi sendiri mempunyai komponen-komponen untuk mencapai tujuan kriptografi. Menurut (Ariyus, 2009), beberapa komponen dalam kriptografi meliputi:

1. Enrkipsi (*Encryption*)

Enkripsi merupakan hal yang sangat penting dalam kriptografi untuk mengamankan sebuah informasi agar pesan yang dikirimkan terjaga kerasahasiaanya. Informasi (yang disebut *plain text*) diubah menjadi serangkaian kode rumit yang sulit diartikan. Enkripsi sendiri bisa diartikan sebagai *chipper* atau kode. Berdasarkan ISO 7498-2, *terminology* yang lebih tepat digunakan untuk menamakan proses ini adalah “*enchiper*”.

1. Dekripsi (*Decryption*)

Dekripsi merupakan kebalikan dari proses enkripsi. Dekripsi yaitu proses mengubah kembali pesan yang telah di enkripsi menjadi pesan aslinya, yang disebut dengan dekripsi pesan. Berdasarkan ISO 7498-2, *terminology* yang lebih tepat untuk menamakan proses ini adalah “*dechiper*”.

1. Kunci (*key*)

Kunci yang dimaksud disini adalah kunci atau sandi yang digunakan untuk melakukan enkripsi maupun dekripsi. Kunci terbagi menjadi dua bagian, yaitu kunci *private* (*private key*) dan kunci *public* (*public* *key*).

1. *Plain text*

*Plain text* disebut juga *cleartext*, yaitu pesan asli yang ditulis atau diketik. *Plain text* inilah yang akan diproses menggunakan algoritme kriptografi agar menjadi *Chiper text.*

1. Pesan

Pesan bisa berupa data atau informasi yang dikirimkan (melalui kurir, saluran komunikasi data dan sebagainya) atau yang disimpan didalam media penyimpanan (kertas, *storage* dan sebagainya).

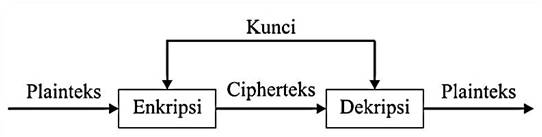
1. *Chiper text*

*Chiper text* merupakan pesan yang dihasilkan dari proses enkripsi. Pesan yang terkandung dalam *Chiper text* ini silit dibaca karena berisi berbagai macam karakter tanpa arti/tidak bermakna.

1. Kriptanalisis (*Cryptanalysis*)

Dapat diartikan sebagai analisis sandi atau suatu ilmu memecahkan *Chiper text* menjadi *plain text* tanpa mengetahui kunci yang digunakan. Pelakunya disebut *cryptanalys* (kriptanalis).

Kriptografi mempunya dua komponen utama yaitu enkripsi dan dekripsi. Selain itu juga dibutuhkan kunci untuk mengubah *plain text* menjadi *Chiper text*, begitu juga sebaliknya. Tanpa kunci, *plain text* tidak bisa melakukan enkripsi pesan menjadi *Chiper text*, juga sebaliknya. Kerahasiaan kunci ini sangatlah penting, apabila kerahasiaanya terbongkar maka isi pesan akan terbongkar. Berikut adalah skema ilustrasi proses enkripsi dan dekripsi.



Gambar 2. 1 Skema Proses Enkripsi dan Dekripsi

Pada gambar 2.4 mengilustrasikan sebuah pesan/*plain text* di enkripsi menggunakan kunci enkripsi sehingga menjadi *Chiper text* yang akan di dekripsi menggunakan kunci dekripsi untuk menghasilkan *plain text* kembali.

## 2.5 Algoritme Grain

Algoritme Grain adalah algoritme yang sengaja didesain atau diciptakan untuk diterapkan pada *hardware* dengan *resource* yang kecil namun dengan implementasi pada sisi *software* yang mudah. Algoritme Grain didesain agar berjalan dengan berorientasi pada setiap bit bukan setiap kata, hal ini bertujuan untuk menurunkan tingkat kompleksitas namun tetap tangguh dalam menangani masalah keamanan data. Algoritme Grain juga memberikan kebebasan pada penggunanya untuk memiliih tingkat kecepatan komputasi tergantung dari *resource* *hardware* yang tersedia. Walaupun Grain diciptakan untuk fokus kepada *hardware* namun grain tetap dapat berjalan baik pada *software* pada umumnya, namun grain tidak dapat dibandingkan dengan algoritme lain yang sengaja dibuat untuk efisiensi pada sisi *software*. Grain diciptakan dengan 2 model *shift* *register* yaitu *linear* *feedback* (LSFR) dan *nonlinear feedback* (NFSR). Kedua *register* memiliki ukuran 80 bits (Hell, et al., 2007)

Algoritme Grain memiliki 3 blok utama yang dinamakan LSFR, NFSR dan blok *filter*. Sebelum setiap *keystream* diubah menjadi *cipher*, LSFR adalah *shift* *register* yang bit masukannya merupakan fungsi *linear* dari state sebelumnya. Pada grain *cipher* isi dari LSFR dapat dinotasikan sebagai berikut :

I,i+1,i+2,….,i+78,i+79 (1)

Fungsi umpan balik dari LSFR, adalah fungsi *polinom* berderajat 80 yang dapat didefinisikan sebagai berikut :

(2)

Dan fungsi update dari LSFR adalah sebagai berikut :

i+80  = i+62 + i+51 + i+38 + i+23 + i+13 + I (3)

Sedangkan isi dari NFSR dapat dinotasikan dengan :

i,i+1, ,i+2,.., ,i+78, ,i+79 (4)

Dan fungsi umpan balik dari NFSR , g(x) dapat didefinisikan sebagai berikut :

(5)

Fungsi update dari NFSR dapat didefinisikan sebagai berikut :

i+80 = i i63 i60 i52 i45 i37

i33i23i21i15i9i

i+63i+60 i+37i+33 i+15i+9

i+60i+52i+45 i+33i+28i+21

i+63i+45i+28i+9i+60i+52i+37i+33

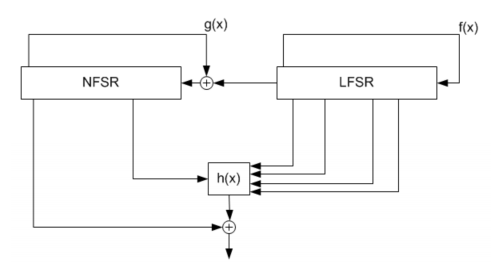
i+63i+60i+21i+15

i+63i+60i+52i+45i+37

i+33i+28i+21i+15i+9

i+52i+45i+37i+33i+28i+21 (6)

Cara kerja dari grain *cipher* secara umum dapat digambarkan dengan diagram sebagai berikut :



Gambar 2. 2 Diagram Umum Grain *Cipher*

Sumber: (Hell Martin,Thomas Johanson dan Willi Meier, 2007)

Isi dari kedua *shift* *register* disebut sebagai kondisi atau state dari *cipher*. Dari *state* yang didapatkan diambil 5 nilai sebagai masukan untuk fungsi filter h(x). Masukan diambil baik dari LSFR dan NSFR. Fungsi tersebut dapat didefinisikan sebagai berikut :

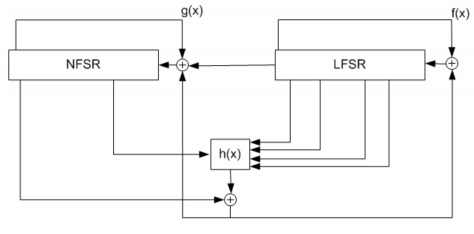
14 +03+23+34+012+023+024+124 +234 (7)

Dimana pengubah 0,1,2,3 dan 4 diambil dari nilai i+3 ,i+25 ,i+46 ,i+64 ,i+63. Keluaran dari fungsi filter ini di-xor (*masked*) dengan bit bi dari NFSR untuk mendapatkan *keystream*.

### 2.5.1 Inisialisasi Kunci

Sebelum setiap *keystream* dibangkitkan, *cipher* harus dinisialisasi dengan kunci dan *IV* (Initial Value). Misalkan bit-bit dari kunci, dilambangkan dengan ki dengan dan bit-bit dari *IV* dilambangkan dengan *IV*i, dengan . Inisialisasi kunci dilakukan dengan cara sebagai berikut:

1. NFSR diisi dengan bit-bit kunci, .
2. 64 bit bertama dari LSFR diisi dengan *IV* , .
3. Sisa bit dari LSFR diisi dengan angka 1 (satu) , .
4. Chiper di-Clock 160 kali.Keluaran dari fungsi filter dikembalikan dan di-xor dengan masukan baik ke LSFR dank e NSFR seperti pada gambar dibawah ini



Gambar 2. 3 Inisialisasi Grain *Cipher*

Sumber : ( Hell et al, 2007)

## 2.6 Algoritme SHA3

Pada tahun 2006, NIST mengadakan kompetisi *hash function* untuk membuat sebuah standar *hash* baru, yaitu SHA-3. SHA-3 dibuat tidak untuk menggantikan SHA-2, dikarenakan belum ada serangan hebat yang terjadi pada SHA-2. NIST membuat SHA-3 dikarenakan kekhawatiran dikarenakan MD5, SHA-0, SHA-1 yang telah berhasil ditembus. Karena itulah NIST mencari algoritme *hash* alternatif yang sangat berbeda dengan algoritme sebelumnya, yaitu SHA-3.

Tahun 2012, *Keccak* menjadi pemenang dalam kompetisi ini. Kemudian pada tahun 2015 mempublikasikan draft FIPS 202 tentang *“SHA-3 Standard: Permutation-Based Hash and Extendable-Output Functions”*. Setelah itu pada tahun 2015 SHA-3 diresmikan sebagai standar baru fungsi hash (Charles, 2015).

Tidak seperti SHA-1, SHA-3 merupakan kelompok dari *spoge function* dikarenakan memiliki ukuran keluaran yang beragam. Jenis-jenis keluaran SHA-3 tersebut yaitu SHA-3 224, SHA-3 256, SHA-3 384, SHA-3 512, SHAKE128 dan SHAKE256*.* Menurut pembuat dari algoritme SHA-3 ini, pada jurnalnya yang berjudul *“Sponge function”, sponge function* menyediakan cara tertentu agar dapat menggeneralisasikan fungsi *hash* dengan hasil keluaran yang beragam (Bertoni, 2011).

Berdasarkan penjelasan dari publikasiresmi FIPS yang berjudul “*SHA-3 Standard: Permutation-Based Hash and Extendable-Output Functions*”, ada beberapa langkah yang harus dilakukan agar menghasilkan hash SHA-3. Langkah-langkah tersebut akan dijelaskan sebagai berikut.

1. Inisialisasi *input* dan *output*

Pada fase ini dilakukan pemilihan jenis algoritme SHA-3 mana yang akan digunakan. Jenis-jenis algoritme SHA-3 tersebut yaitu SHA-3 224, SHA-3 256, SHA-3 384, SHA-3 512, SHAKE128 dan SHAKE256. Parameter yang dibutuhkan dari fase ini yaitu *message* masukan dari pengguna, misalnya “plainttext”. Masukan tersebut akan diproses, setelah diproses maka akan dihasilkanlah *digest* dari *message* tadi.

1. *Absorbing*

Pada fase ini, *block state* dari *plain text* yang telah diproses pada fase sebelumnya tadi akan dimasukkan kedalam algoritme dan kemudian diproses lagi. Proses ini juga melibatkan fungsi permutasi Keccak.

1. *Squeezing*

Fase ini merupakan fase proses perhitungan untuk *digest* yang akan dihasilkan. Panjang yang dihasilkan akan sesuai dengan yang dipilih pengguna pada fase inisialisasi. Fase ini juga melibatkan permutasi Keccak.

1. Permutasi Keccak

Permutasi Keccak merupakan proses paling utama dari algoritme ini. Semua fase yang dilalui harus melalui fungsi ini dulu. Permutasi Keccak dibagi menjadi 5 tahapan, yaitu ρ *(Rho),* θ *(Theta)*, χ *(Chi)*, π *(Phi)*, dan ι *(Iota)*. Tahapan-tahapan tersebut memiliki fungsi masing-masing, dan fungsi tadi telah ditetapkan oleh NIST. Fungsi-fungsi ini dilihat merujuk dari publikasi resmi FIPS 202 oleh NIST.

Setiap tahapan akan dilakukan perulangan sebanyak 25 kali.

**θ *(Theta)***

Operasi *Theta* bersifat linier. Operasi *Theta* dapat digambarkan sebagai penambahan pada setiap bit A[X][Y][Z] pada penjumlahan bit dari A[X-1][.][Z] dan A[X+1][.][Z-1]. Operasi *Theta* hanya meng-XOR-kan 11 bit menjadi satu, sehingga setiap bit berpengaruh pada 11 bit lainnya. Operasi yang dilakukan pada Theta dijelaskan lewat *pseudocode* sebagai berikut.

C[x] = A[x,0]XOR A[x,1] XOR A[x,2] XOR A[x,3] XOR A[x,4] , x = 0,1,2,3,4 (8)

D[x] = C[x-1] XOR rot(C[x+1],1) , x = 0,1,2,3,4 (9)

A[x,y] = A[x,y] XOR D[x] , x,y = 0,1,2,3,4 (10)

**ρ *(Rho)* dan π *(Phi)***

Pada operasi ini akan dilakukan pemetaan terhadap *list array* tadi. Operasi ini juga bersifat linier, dengan invers berupa penggeseran ulang yang berkebalikan dengan penggeseran sebelumnya. Operasi tersebut dapat digambarkan dengan *pseudocode* dibawah ini.

B[y,2x+3y] = rot(A[x,y], r[x,y]) , x,y = 0,1,2,3,4

**χ *(Chi)***

Operasi *Chi* merupakan satu-satunya operasi pada permutasi Keccak yang pemetaannya tidak linier. Operasi dapat dilihat sebagai aplikasi operasi *S-box* untuk 5-bit baris. Pada operasi ini dilakukan manipulasi terhadap *list array* B dari operasi sebelumnya yaitu *Rho* dan *Phi*. Kemudian hasil dari manipulasi tersebut dikembalikan lagi ke list array A. *Pseudocode* untuk operasi ini adalah sebagai berikut:

A[x,y] = B[x,y]XOR ((¯B[x+1,y])^B[x+2,y]) , x,y = 0,1,2,3,4 (11)

**ι *(Iota)***

Dalam operasi ini dilakukan penambahan konstanta ke *list array* pada lokasi [0,0] dari list array A. *Pseudocode* dari operasi tersebut yaitu sebagai berikut:

A[0,0] = A[0,0]XOR RC[i]

## 2.7 Perangkat Bergerak

Perkembangan terknologi dari tahun ke tahun memperlihatkan suatu peningkatan yang signifikan, dimana saat ini teknologi berkembang menjadi suatu perangkat perangkat yang lebih ringkas ataupun lebih kecil daripada sebelumnya, namun memiliki fungsi yang sama, bahkan bisa lebih canggih daripada sebelumnya. Perangkat tersebut biasa disebut sebagai perangkat bergerak, karena kita bisa membawa dan menggunakan perangkat tersebut dimananpun dan kemanapun yang kita ingingkan

Perangkat bergerak merupakan sebuah perangkat kecil yang kemampuan komputasinya terbatas karena beberapa faktor. Perangkat bergerak biasa disebut sebagai perangkat genggam karena kemudahanya dibawa kemana-mana. Perangkat bergerak pada umumnya terdiri dari dua bagian, yang pertama layar *display* sebagai perangkat keluaran dan *key*board atau layar sentuh sebagai masukan. Pada perangkat telepon pintar (*Smartphone*), umumnya perangkat layar visual selain berfungsi sebagai perangkat keluaran, juga berfungsi sebagai perangkat masukan, karena sifatnya yang memiliki layar sentuh pada keseluruhan tampilanya. (Zaki, 2008)

Karena fungsinya yang bervariasi, maka sebuah organisasi T3B dan *DuPont Global Mobility* merumuskan defisini standar dari perangkat bergerak (Zaki,2008), yaitu :

1. Limited Data *Mobile* *Device* : Perangkat yang ukuran layarnya kecil, umumnya layar hanya menampilkan teks, layanan data hanya terbatas ke SMS dan WAP. Contoh perangkat ini adalah telepon seluler.
2. Basic Data *Mobile* *Device* : Perangkat yang ukuran layarnya menengah . Memiliki navigasi menggunakan menu atau ikon. Layanan yang ditawarkan antara lain surel (email), daftar alamat (Kontak), SMS, dan *Web Browser*. Contoh perangkat ini adalah telepon pintar (*Smartphone*)
3. Enhanced Data *Mobile* *Device* : Perangkat yang ukuran layarnya besar, biasanya menggunakan pena stylus untuk layar sentuhnya, dan memiliki fitur-fitur layanan dasar yang dimiliki oleh piranti sebelumnya ditambah adanya kemampuan untuk menambah berbagai aplikasi seperti *Microsoft Office* dan portal internet. Contoh perangkat ini adalah *Pocket PC* dan Tablet.

### 2.7.1 Aplikasi Perangkat Bergerak

Aplikasi perangkat bergerak (*mobile*) adalah suatu aplikasi yang dibuat secara khusus untuk berjalan pada *mobile* *device*. Aplikasi *mobile* pada umumnya dikelompokan berdasarkan platform, beberapa kategori platform tersebut adalah:

* *Android*
* iPhone
* Windows *Mobile*
* Blackberry

Aplikasi *mobile* sendiri secara spesifik juga dikembangkan berdasarkan masing-masing platform sesuai dengan kebutuhanya (Native, 2014)

1. *Native Application*

Aplikasi yang dibuat dan dipasangkan langsung ke dalam *device* menggunakan bahasa pemrograman khusus untuk membuat aplikasi tersebut. Contohnya adalah untuk membuat aplikasi pada platform *Android* digunakan Java dan *Software* *Development kit* (SDK), juga platform iOS yang menggunakan *Objective*-C dan SDK iOS

1. *Mobile* Web Application

*Mobile Web Application* merupakan aplikasi yang membutuhkan browser untuk menjalankanya, biasanya dibuat menggunakan bahasa pemrograman web seperti PHP & HTML 5.

1. Hybrid Application

*Hybrid Application* meruapakan gabungan dari aplikasi native dan juga web. Dapat dibuat menggunakan bahasa pemrograman web yang digabung dengan bahasa pemgorgraman yang digunakan untuk membuat aplikasi pada *device* yang dituju.

## 2.8 *Client* *Server*

*Client Server* merupakan salah satu arsitektur dalam jaringan komputer. Arsitektur ini merupakan model konektivitas pada jaringan yang mengenal adanya device yang disebut *server* dan disebut sebagai *client*, dimana masing-masing memiliki fungsi yang berbeda satu sama lain. Prinsip kerja *client* *server* sangat sederhana yaitu *client* dapat memberikan *request* kepada *server* dan *server* akan menangani *request* tersebut lalu memberikan hasilnya kepada client. *Server* hanya akan bekerja ketika mendapatkan *request* dari *client*.

### 2.8.1 Sistem *Client Server*

Seperti namanya Sistem *Client* dan *Server* terdiri atas dua komponen (mesin) utama, yaitu *Client* dan *Server*. Setiap aktifitas yang dilakukan oleh *user* umumnya akan lebih dahulu ditangani oleh *client*. *Client* menangani proses yang menjadi tanggung jawabnya. Namun jika terdapat sebuah proses yang harus melibatkan data yang tersimpan pada basis data yang terletak di *server* contohnya, barulah *client* mengadakan hubungan dengan *server* dan memberikan *request* terhadap *server*. *Client* akan megirimkan *request* kepada *server*. Selanjutnya *server* yang menerima *request* tersebut akan menjalankan memproses request *t*ersebut dan hasilnya akan di kirimkan kembali ke *client*. Dengan begitu, transfer datanya jauh lebih efisien.

# BAB 3 METODOLOGI

Bab ini menjelaskan metode yang digunakan dalam penelitian, yaitu studi *literature*, analisis kebutuhan, perancangan *system*, implementasi, pengujian, dan analisis, pengambilan kesimpulan dan saran. Runtutan pengerjaan peneltian dapat dilihat pada diagram alir berikut.

Implementasi

Pengujian dan Analisis

Analisis Kebutuhan

Perancangan Sistem

Pengambilan Kesimpulan & Saran

**Gambar 3.1** Diagram Alir Penelitian

## 3.1 Analisis Kebutuhan

Analisis kebutuhan merupakan kegiatan untuk mengetahui keubutuhan apa saja yang harus terpenuhi pada penelitian ini. Analisis kebutuhan dalam penelitian ini dilakukan dengan mengidentifikasi kebutuhan dari sistem keamanan aplikasi *client* pemilu *online* dengan Algoritme Grain dan SHA-3.

Selain menganalisis kebutuhan dari sisi aplikasi pemilu *online* dilakukan juga anaisis kebutuhan yang dibutuhkan oleh algoritme grain dan SHA-3 agar keduanya dapat berjalan dengan baik. Pada algoritma grain dilakukan analisis kebutuhan terkait dengan nilai *variable input* yang dibutuhkan oleh algoritma grain, sedangkan pada SHA-3 dilakukan analisis terhadap jenis SHA-3 yang akan diterapkan pada sistem aplikasi pemilu *online*.

## 3.2 Perancangan Sistem

Perancangan sistem dilakukan setelah tahap analisis kebutuhan selesai dilakukan. Terdapat dua langkah dalam perancangan sistem yaitu perancangan diagram alir serta perancangan desain tampilan dari aplikasi.

Perancangan diagram alir sistem keamanan aplikasi dilakukan untuk menggambarkan proses perlindungan terhadap informasi pemilu *online* yang dikirimkan oleh peserta pemilu dan diterima oleh *server* melalui proses enkripsi dan hash menggunakan algoritme kriptografi Grain dan SHA-3.

Sedangkan perancangan desain tampilan aplikasi dilakukan untuk memudahkan dalam melakukan pembuatan *user interface* pada android studio, sehingga pada saat melakukan pengkodean, penulis sudah tahu bahwa aplikasi yang dibuat harus memiliki tampilan seperti apa.

## 3.3 Implementasi

Implementasi dilakukan untuk merealisasikan model yang telah dirancang pada proses sebelumnya, menjadi sebuah aplikasi yang dapat memenuhi kebutuhan atau *requirement* dari aplikasi. Proses implementasi dimulai dengan menjabarkan spesifikasi singkat *environment* dari *system* yang akan dibangun. Spesifikasi ini terdiri dari spesifikasi *hardware* dan *software* yang dibutuhkan agar aplikasi *system* keamanan pemilu *online* dapat berjalan dengan normal. Langkah selanjutnya dilanjutkan ke pembuatan keaman ekripsi terhadap informasi pemilu peserta memanfaatkan algoritme kriptografi Grain dan SHA-3 pada perangkat bergerak menggunakan bahasa pemrograman Java. Sedangkan database *server* yang digunakan dalam pelaksaan pemilu *online* ini dibangun dengan menggunakan *service firebase* yang telah disediakan oleh Google.

## 3.4 Pengujian dan Analisis

Pengujian perangkat lunak dilakukan agar dapat diketahui apakah perangkat lunak dapat berjalan optimal sesuai dengan spesifikasi, tujuan, dan kebutuhan yang telat dirancang sebelumnya. Strategi pengujian perangkat lunak yang akan digunakan antara lain :

1. Pengujian Kerahasiaan (*Confidentiality*) dilakukan untuk memastikan data pilihan peserta pemilu *online* terjaga. Dengan metode Grain maka *system* keamanan aplikasi akan memberi perlindungan fisik pada data pilihan *user* yang terkirim saat pemilu *online* berlangsung, data tersebut diubah menjadi bentuk *cipher text* yang sulit dimengerti isinya. Dengan metode ini juga akan meminimalkan celah bagi pihak lain untuk mengetahui isi informasi pemilih yang bukan termasuk kewenanganya saat akan dilakukan *sniffing* terhadap lalu lintas paket data pemilu *online* dalam suatu jaringan.
2. Pengujian Integritas (*Integrity*) dilakukan untuk mengetahui keutuhan data pilihan peserta pemilu *online* yang telah tersimpan di dalam *database* setelah melaksakan pemilu *online*. Data tersebut berupa pilihan paslon pengguna yang telah diubah dari bentuk *plain text* menjadi *chiper text* dengan teknik enkripsi memanfaatkan algoritme kriptografi grain dan *cipher text* tersebut akan dijadikan sebagai masukan untuk algoritme SHA-3 dengan tujuan untuk mensegel data tersebut agar tidak dapat dirubah-rubah dan tetap utuh selain untuk merahasiakan isinya. SHA-3 disini berfungsi sebagai *hashing* agar data yang sudah dirubah menjadi *digest text* tidak dapat diubah kembali menjadi *plain text*.

3. Pegujian Waktu Pemrosesan dilakukan untuk mengetahui waktu pemrosesan yang diperlukan oleh aplikasi pemilu berbasis android ketika menerapkan algoritma grain dan SHA-3 dibandingkan ketika tidak menerapkan kedua algoritma tersebut. Setelah itu hasil dari pengujian ini akan dilakukan pengujian *Independent* T*-Test* untuk mengetahui apakah dari kedua rata-rata pengujian ini memiliki perbedaan yang signifikan atau tidak.

4. Pengujian *Avalanche Effect* dilakukan untuk mengetahui seberapa banyak karakter yang berubah ketika input yang dimasukan hanya berbeda 1 karakter pada algoritma grain, SHA-3 dan juga ketika kedua algoritma tersebut berjalan secara *linear*.

5. Pengujian Skema Pembentukan Kunci dilakukan untuk mengetahui apakah sistem dapat membuat kunci secara dinamis memanfaatkan kombinasi dari NIK dan *password* sebagai inputan key untuk algoritma grain dalam mengenkripsi data.

## 3.5 Pengambilan Kesimpulan dan Saran

Pengambilan kesimpulan dan saran merupakan tahap terakhir pada penelitian ini. Kesimpulan diambil dari hasil pengujian terhadap aplikasi yang telah dibangun. Tahap terakhir penulisan adalah saran yang dimaksudkan untuk memperbaiki kekurangan yang terjadi dan menyempurnakan penulisan serta mengembangkan penelitian lebih lanjut.

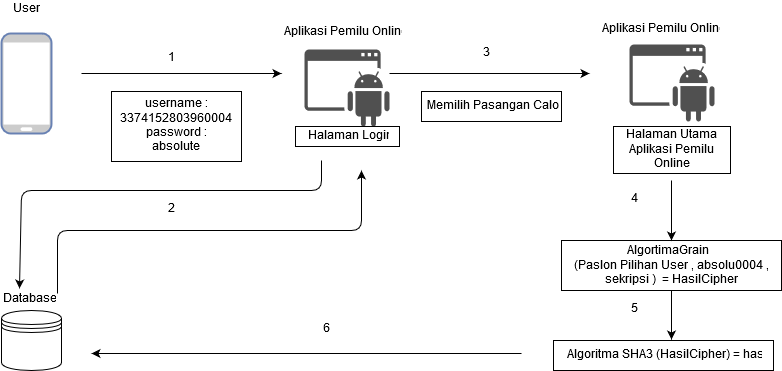
# BAB 4 ANALISIS DAN PERANCANGAN

## 4.1 Analisis Kebutuhan

Pengertian dari analisis ialah penguraian dari suatu pembahasan, dalam hal ini analisis mengenai perancangan sistem pemilu *online* yang dilengkapi dengan algoritmegrain sebagai proses enkripsi data dan juga algoritme SHA-3 sebagai proses *hashing* data yang berguna untuk mengetahui apa saja yang dapat dijadikan isi dari perancangan yang akan dibuat.

### 4.1.1 Analisis Permasalahan

Dalam penelitian yang sedang dibahas yaitu perancangan sistem pemilu *online* yang dilengkapi dengan algoritme grain sebagai proses enkripsi data dan juga algoritme SHA-3 sebagai proses *hashing* data. Analisis dari permasalahan yang sedang dibahas pada penelitian ini akan dijelaskan pada Gambar 4.1



Gambar 4. 1 Skema Global Aplikasi Pemilu *Online*

Penjelasan skema global Aplikasi Pemilu *Online* pada Gambar 4.1 di atas adalah sebagai berikut.

1. Pengguna melakukan *login* dengan memasukan *username* berupa Nomor Induk Kewarganegaraan (NIK) dan memasukan *password* agar dapat masuk ke dalam *system*.

2. Kemudian *username* dan *password* tersebut akan dicocokan pada data *username* dan *password* yang sudah terdaftar pada database, jika terjadi kecocokan dengan salah satu data maka *user* akan masuk kedalam halaman utama Aplikasi Pemilu *Online*.

3. *User* melakukan pemilihan terhadap pasangan calon yang ia inginkan pada Aplkasi Pemilu *Online*.

4. Data pilihan pasangan calon yang dipilih oleh *user* akan di enkripsi dengan algoritme *grain*. Algoritme *grain* memerlukan 3 buah *input*an, pada kasus ini ketiga *input*an tersebut adalah data pilihan pasangan calon yang dipilih oleh *user*, 8 *digit* awal *password* diikuti 2 *digit* akhir *username* sebagai *key* dan kata “sekripsi” yang secara *defalut* telah diatur oleh penulis sebagai *Initial Value* (*IV*).

5. Keluaran dari proses enkripsi pada tahapan sebelumnya yaitu berupa *cipher text* akan dilalui proses *hashing* dengan menggunakan algoritme *hashing* SHA-3.

6. Hasil dari proses *hashing* yang berupa *digest text* akan di masukan kedalam *database*.

Dengan skema yang diharapkan diatas, diharapkan tidak ada kemugkinan terjadinya upaya pelemahan tingkat keamanan oleh pihak tidak berwenang dari sisi aplikas pada saat dilakukanya aktifitas pemilu *online*.

*Plain Text*

Inisialisasi Kunci

Enrkipsi

*Ciphertext*

Permutasi Grain

Permutasi Grain

Gambar 4. 2 Skema Global Algoritme Grain

Penjelasan skema global dari algoritme grain pada Gambar 4.2 adalah sebagai berikut.

1. Pada proses inisialisasi kunci diperlukan 2 buah masukan yaitu *Key* dan *Initial Value* (*IV*). Lalu kedua data tersebut dirubah menjadi bilangan *binary* dan dimasukan kedalam blok fungsi NFSR untuk *key* dan LFSR untuk *IV* yang dimana masing-masing blok fungsi memiliki daya tampung maksimal 80 bit. Proses ini dilakukan permutasi *grain* dan di *clock* sebanyak 160 kali.

2. Setelah itu data yang akan di enkripsi di rubah dahulu menjadi bilangan *binary* dan dihitung panjangnya. Lalu dilakukan permutasi *Grain* kembali sebanyak panjang dari data yang akan dienkripsi dengan nilai blok fungsi NFSR dan LFSR yang sudah diacak sebanyak 160 kali pada tahap sebelumnya. Pada tahap ini setiap *clock* yang dilakukan akan diambil nilainya dan dimasukan kedalam suatu variable sehingga nilai dari *variable* tersebut memiliki panjang yang sama dengan panjang data yang akan di enkripsi. Nilai *variable* tersebut lalu dilakukan perhitungan XOR dengan nilai *binary* dari data yang akan dienkripsi tadi sehingga terbentuklah *Cipher text*.

Gambar 4. 3 Skema Global Algoritma SHA-3

*Cipher Text*

Inisialisasi *Input*

*Absorbing*

*Squeezing*

Inisialisasi *Output*

*Hashed* *Text*

Permutasi *Keccak*

Permutasi *Keccak*

Penjelasan skema global dari algoritme SHA-3 pada Gambar 4.3 adalah sebagai berikut.

* + - 1. *Ciphert text* yang merupakan keluaran dari proses *hashing* yang telah dilakukan sebelumnya akan melalui fase inisialisasi, yaitu fase untuk menentukan algoritme SHA-3 jenis apa yang akan digunakan, dalam kasus ini akan digunakan algoritme SHA-3 224. Kemudian *cipher text* tersebut dimasukkan ke dalam *block state array. Block state array* merupakan sebuah *array* 3 dimensi yang dibutuhkan untuk proses *hashing* pada algoritme SHA-3.
      2. Fase selanjutnya ialah fase *absorbing,* yaitu untuk fase memasukkan *block state array* tadi pada algoritme permutasi Keccak, yang terdiri dari 5 tahapan, yaitu θ *(Theta)*, ρ *(Rho)*, π *(Phi)*, χ *(Chi)* dan ι *(Iota)*.
      3. Kemudian hasil dari fase sebelumnya yaitu *absorbing* akan dilanjutkan dengan fase *squeezing,* yaitu fase pengeluaran dari *block state* yang sudah di lakukan pada fase *absorbing* tadi. Pada fase ini juga dilakukan permutasi *Keccak* pada *array* tersebut seperti yang sudah dijelaskan pada nomor 2.
      4. Fase terakhir ialahmelakukan inisialisasi lagi terhadap hasil dari fase-fase sebelumnya agar sesuai dengan format jenis algoritme SHA-3 yang digunakan. Kemudian dihasilkanlah *digest text* dari algoritme *hash* SHA-3 ini.

### 4.1.2 Analisis Data

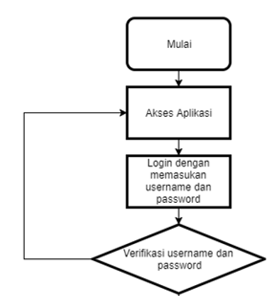
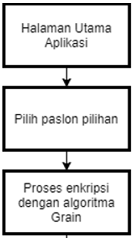
Analisis data merupakan tahapan dimana dilakukannya analisis terhadap data-data apa saja yang diolah dalam *system*. Pada aplikasi ini, data yang akan di lakukan proses enkripsi dan *hashing* hanya data polling yang dilakukan oleh *user* berupa informasi pasangan calon. Untuk masuk kedalam *system*, *user* diwajibkan memasukan dua data yaitu *username* berupa Nomor Induk Kependudukan (NIK) dan juga *password* yang sudah terdata pada *database* sebelumnya.

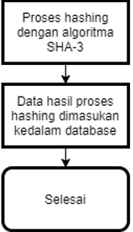
### 4.1.3 Analisis Keamanan Data

Dalam *system* pemilu *online* ini, tedapat data sensitif yang ditransmisikan dari *gadget* pengguna menuju *server* pemilu *online*. Data-data tersebut ialah *username*, *password* dan juga paslon pilihan pengguna. Pemilu di Indonesia bersifat Luber dan Jurdil yang berarti Langsung, Bebas, Rahasia, Jujur dan Adil maka penulis memutuskan untuk melakukan proses enkripsi dan *hashing* pada data polling dari pengguna aplikasi pemilu *online*. Penulis memilih data tersebut yang diamankan karena akan memenuhi sifat Rahasia dan Bebas. Ketika dilakukan enkripsi data menjadi rahasia karena tidak dapat dibaca oleh orang lain dan ketika dilakukan *hashing* maka data tidak dapat dirubah oleh pihak lain dan menjadikan suatu kebebasan memilih untuk masing – masing asing pengguna aplikasi pemilu *online*.

## 4.2 Perancangan Sistem

### 4.2.1 Flowchart Sistem





Gambar 4. 4 Flowchart Sistem Aplikasi Pemilu *Online*

Prosedur *flowchart* pada Gambar merupakan gambaran Aplikasi Pemilu *Online* dengan menerapkan algoritme *Grain* sebagai proses enrkipsi data dan algoritme SHA-3 sebagai proses *hashing* data sebelum data pasangan calon pilihan *user* dimasukan kedalam database. Dibawah ini adalah prosedur kerja dari *flowchart* pada Gambar 4.4.

* + 1. Mulai

1. Masuk ke halaman awal aplikasi
2. Melakukan *login* dengan memasukan *username* berupa Nomor Induk Kewarganegaraan (NIK) dan memasukan *password*.
3. *Username* dan *password* yang sudah dimasukan pada tahapan sebelumnya akan dicocokan pada data *username* dan *password* yang sudah tersimpan didatabase sebagai proses verifikasi, jika terjadi kecocokan maka *user* akan masuk kedalam halaman utama Aplikasi Pemilu *Online*, jika tidak maka *user* akan tetap berada pada halaman awal.
4. Selanjutnya *user* akan diminta untuk memilih pasangan calon pilihanya pada halaman utama dan menekan tombol “OK”.
5. Sebelum data pasangan calon pilihan *user* di masukan kedalam *database*, akan dilakukan proses ekripsi terlebih dahulu menggunakan algoritme Grain agar data tersebut tidak dapat dibaca oleh pihak yang tidak berwenang dan guna memberikan asas *Confidentiality* kepada *user*.
6. Keluaran dari proses enkripsi pada tahapan sebelumnya akan melalui proses *hashing* mengunakan algoritme hash SHA-3 agar data tersebut tidak dapat diubah oleh siapapun dan guna memberikan asas *integrity* kepada *user*.
7. *Digest text* atau keluaran dari proses *hashing* pada tahapan sebelumnya akan dimasukan kedalam *database*.
8. Selesai.

### 4.3 Desain Tampilan

Fungsi dari perancangan desain tampilan ini adalah untuk mempermudah dalam proses implementasi, dan juga agar sistem yang dibuat lebih efektif sesuai dengan harapan. Berikut ini adalah rancangan desain tampilan.

1. Desain tampilan halaman *login*

APLIKASI PEMILU *ONLINE*

1

*USERNAME*

2

*PASSWORD*

3

**MASUK**

Gambar 4. 5 Desain Tampilan Halaman Login

Berikut ini merupakan keterangan untuk desain tampilan halaman *login* yang terdapat pada Gambar 4.5

* 1. Form masukan untuk *username*
  2. Form masukan untuk *password*
  3. Tombol Masuk untuk proses autentikasi kecocokan *username* dan *password* dengan data yang sudah terdapat dalam *database*

2. Desain tampilan halaman pemilihan pasangan calon

PILIH PASANGAN CALON PILIHAN ANDA

**1**

**2**

**PILIH**

**3**

Gambar 4. 6 Desain Tampilan Halaman Calon

Berikut ini merupakan keterangan untuk desain tampilan halaman pemilihan pasangan calon yang terdapat pada Gambar

* + 1. Foto dari masing-masing pasangan calon yang tersedia
    2. *Radio button* untuk memilih pasangan calon diatas *radio button* tersebut
    3. Tombol Pilih untuk melakukan verifikasi pemilihan pasangan calon dan mengirimkan data tersebut ke *database*

3. Desain tampilan halaman akhir

**TERIMA KASIH TELAH MEMILIH**

PILIHAN ANDA TELAH TERCATAT DALAM DATABASE

**1**

**SELESAI**

Gambar 4. 7 Desain Tampilan Halaman Akhir

Berikut ini merupakan keterangan untuk desain tampilan halaman pemilihan pasangan calon yang terdapat pada Gambar

* + - 1. Tombol selesai untuk menyudahi penggunaan aplikasi dan melakukan *logout* otomatis setelah ditekan

# BAB 5 IMPLEMENTASI DAN PENGUJIAN

Implementasi merupakan tahapan penerapan terhadap sistem yang telah dirancang sebelumnya. Dalam penelitian ini, penulis menggunakan bahasa pemrograman Java. Berdasarkan hasil analisis dan perancangan yang sudah dilakukan, akan dilakukan pembuatan aplikasi pemilu *online* berbasis *android* yang mengimplementasi algoritme *Grain* dan SHA-3 dan menggunakan *service firebase* sebagai database *online* untuk menyimpan hasil dari pilihan *user*.

Pada bagian pengujian, akan dilakukan pengujian dengan memberikan beberapa jenis serangan terhadap sistem yang telah dibuat untuk memastikan tingkat *Confidentiality* dan *Integrity* *user* dari aplikasi yang telah dibuat sebelumnya.

Tabel 5.1 Tabel Spesifikasi Perangkat Lunak dan Perangkat Keras

|  |  |  |
| --- | --- | --- |
| No. | Jenis Perangkat | Komponen |
| 1. | Perangkat Keras | a. Intel® Core™ i7-4710 HQ CPU @ 2.50GHz  b. RAM 8.00 GB DDR 3  c. Harddisk dengan kapasistas 1 TB  d. VGA Nvidia GTX850 4GB  e. Qualcomm Atheros AR9485WB-EG Wireless Network Adapter  f. 4 GB RAM pada *Smartphone*  g. Exynos 8895 Octa |
| 2. | Perangkat Lunak | a. Windows 8.1 Single Language 64-bit  b. *Android* Studio  b. *Netbeans ide* 8.0.1  c. Firebase Console  d. Google Chrome  e. *Android* 7.0 (Nougat) |

## 5.1 Implementasi

Pada proses ini akan diterapkan fungsi algoritme Grain yang penulis telah buat sebelumnya dan fungsi algoritme SHA-3 dengan menggunakan *library* yang diunduh pada situs [*https://github.com/romus/sha*](https://github.com/romus/sha) pada aplikasi pemilu berbasis *android*. Proses ini digunakan sebagai perlindungan data pilihan *user* pada aplikasi pemilu *online* yang akan disimpan pada *database* agar tidak dapat dibaca dan dirubah oleh pihak lain.

### 5.**1.1 Algoritme Grain**

#### 5.1.1.1 LFSR (*Linear* Feedback *Shift* *Register*)

LFSR pada grain berkapasitas 80 bit dengan ketentuan 64 bit awal diisi oleh nilai binary dari initial value dan 16 sisa bit terakhir diisi dengan nilai default 1. Namun jika nilai *input* *IV* setelah dirubah menjadi bentuk *binary* tidak berjumlah 64 bit, maka bit sisanya akan diisi dengan nilai 0. Berikut *source code* yang menjelaskan proses kerja LFSR :

Tabel 5. 2 *Source Code* Inisialisasi LFSR

|  |  |
| --- | --- |
| Baris Ke - | *Source* *Code* |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14 | public void init(String *IV*,String *key*){  String *IV*\_result = this.hexToBinary(*IV*);  *System*.out.println("*IV* : " +*IV*\_result);  for (int i = 0; i < *IV*\_result.length(); i++) {  lfsr[i] = Byte.parseByte(String.valueOf(*IV*\_result.charAt(i)));  }  if (*IV*\_result.length()<64) {  for (int i = *IV*\_result.length(); i < 64; i++) {  lfsr[i]=0;  }    }  for (int i = 64; i < 80; i++) {  lfsr[i] =1 ; |

*Code* 1 mejelaskan tentang proses pengisian data pada LFSR menggunakan nilai *binary* dari *initial value* (*IV*) untung 64 bit awal dan juga pengisian nilai 16 bit terakhir menggunakan nilai default yaitu 1. Pada LFSR terdapat rumus f(x) sebagai pemrosesan aritmatika. Berikut *source code* aritmatika.

Tabel 5. 3 *Source Code* Aritmatika LFSR

|  |  |
| --- | --- |
| Baris Ke - | *Source* *Code* |
| 1  2  3  4  5  6  7 | public void init(String *IV*,String *key*){  public void fx(){  int[] data = this.lfsr;    this.fx = (byte) (data[62] ^ data [51] ^ data [38] ^ data [23]^ data [13]^ data [0] );  } |

*Code* 2 menjelaskan rumus fx yang digunakan setiap dilakukan 1 kali clock pada LFSR yang menghasilkan suatu nilai *output* baru yang nantinya nilai tersebut akan diproses lebih lanjut.

#### 5.1.1.2 NFSR (Non-*Linear* Feedback *Shift* *Register*)

Sama dengan LFSR, NFSR pada grain memiliki kapasitas 80 bit namun untuk pengisian nilai setiap blocknya diisi dengan nilai dari *key* yang telah dirubah terlebih dahulu menjadi bentuk binary, namun jika nilai *input* *key* setelah dirubah menjadi bentuk *binary* tidak berjumlah 80 bit, maka bit sisanya akan diisi dengan nilai 0. Berikut *source* *code* yang menjelaskan proses kerja LFSR :

Tabel 5. 4 *Source Code* Inisialisasi NFSR

|  |  |
| --- | --- |
| Baris Ke - | *Source* *Code* |
| 1  2  3  4  5  6  7  8  9  10  11  12 | public void init(String *IV*,String *key*){  String *key*\_result = this.hexToBinary(*key*);  //memasukan *key*\_result kedalam array  for (int j = 0; j < *key*\_result.length(); j++) {  nfsr[j] = Byte.parseByte(String.valueOf(*key*\_result.charAt(j)));  }  if (*key*\_result.length()<80) {  for (int j = *key*\_result.length(); j < 80; j++) {  nfsr[j]=0;  }  }  } |

*Code* 3 menjelaskan tentang proses pengisian data pada NFSR menggunakan nilai binary dari *key*. Pada NFSR terdapat rumus g(x) sebagai pemrosesan aritmatika. Berikut *source code* aritmatika.

Tabel 5. 5 *Source Code* Aritmatika NFSR

|  |  |
| --- | --- |
| Baris Ke - | *Source* *Code* |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21 | public int gx(){  int[] data = this.nfsr;  int[] data2 = this.lfsr;  this.gx = (byte)(  data2[0] ^ data[62] ^ data[60]  ^data[52] ^ data[45] ^ data[37]  ^data[33] ^ data[28] ^ data[21]  ^data[14] ^ data[9] ^ data[0]  ^ (data[63] & data[60])^ (data[37] & data[33])  ^ (data[15] & data[9]) ^ (data[60] & data[52] & data[45])  ^ (data[33] & data[28] & data[21])  ^ (data[63] & data[45] & data[28] & data[9])  ^ (data[60] & data[52] & data[37] & data[33])  ^ (data[63] & data[60] & data[21] & data[15])  ^ (data[63] & data[60] & data[52] & data[45] & data[37])  ^ (data[33] & data[28] & data[21] & data[15] & data[9])  ^ (data[52] & data[45] & data[37] & data[33] & data [28] & data[21])  );  return this.gx ;  } |

*Code* 4 menjelaskan rumus gx yang digunakan setiap dilakukan 1 kali clock pada NFSR yang menghasilkan suatu nilai *output* baru yang nantinya nilai tersebut akan diproses lebih lanjut.

#### 5.1.13 Filter *Function*

Selain memiliki *block* LFSR dan NFSR, grain juga memiliki *block* lainya yang bernama Filter *Function*. Filter *function* memiliki rumus h(x) yang yang mengambil beberapa nilai dari LFSR dan juga 1 nilai dari NFSR. Filter *function* akan mengeluarkan nilai keluaran yang pada proses selanjutnya akan di XOR kan dengan hasil dari Z *Function*.

Tabel 5. 6 *Source Code* Filter *Function*

|  |  |
| --- | --- |
| Baris Ke - | *Source* *Code* |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17 | public int hx(){  int[] data = this.nfsr;  int[] data2 = this.lfsr;  int xor;    int x0 = this.lfsr[3];  int x1 = this.lfsr[25];  int x2 = this.lfsr[46];  int x3 = this.lfsr[64];  int x4 = this.nfsr[63];  hx = (x1 ^ x4 ^ (x0 & x3) ^ (x2 & x3) ^ (x3 & x4) ^ (x0 & x1 & x2 )  ^ (x0 & x2 & x3) ^ (x0 & x2 & x4) ^ (x1 & x2 & x4) ^ (x2 & x3 & x4));    return hx;    } |

*Code* 5 mejelaskan tentang proses filter *function* mengambil beberapa nilai dari LFSR dan NFSR yang selanjutkan nilai tersebut akan dilakukan perhitungan dengan rumus h(x) dan menghasilkan suatu nilai keluaran.

#### 5.1.1.4 Z *Function*

Z *Function* adalah fungsi terakhir yang dimiliki oleh algoritme grain. Z *function* mengambil beberapa nilai dari NFSR lalu diolah dengan hasil dari rumus h(x) pada tahapan sebelumnya sehingga menghasilkan *keystream*. *Source code* mengenai Z *function* akan dijelaskan pada *code* 6

Tabel 5. 7 *Source Code* Z *Function*

|  |  |
| --- | --- |
| Baris Ke - | *Source* *Code* |
| 1  2  3  4  5  6  7  8  9  10  11  12  13 | public int z(){    int z0 = this.nfsr[1];  int z1 = this.nfsr[2];  int z2 = this.nfsr[4];  int z3 = this.nfsr[10];  int z4 = this.nfsr[31];  int z5 = this.nfsr[43];  int z6 = this.nfsr[56];    z = z0 ^ z1 ^ z2 ^ z3 ^ z4 ^ z5 ^ z6;  return z;  } |

#### 5.1.1.4 Proses Inisialisasi Kunci

Proses inisialisasi kunci pada grain berfungsi untuk mengacak nilai *state* pada LFSR dan NFSR dengan alur dan rumus yang telah ditentukan sebanyak 160 kali *clocking*. Berikut merupakan *source code* dari fungsi inisialisasi kunci.

Tabel 5. 8 *Source Code* Inisialisasi Kunci

|  |  |  |
| --- | --- | --- |
| Baris Ke - | | *Source* *Code* |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24 | public void inisialisasikunci(){  // initTest();  int new\_nfsr;  int new\_lfsr;  int new\_filter;    for (int i = 0; i < 160; i++) {  fx();  gx();  hx();  z();  new\_filter = hx ^ z;  new\_nfsr = gx ^ new\_filter;  new\_lfsr = fx ^ new\_filter;  for (int j = 0; j < 159; j++) {  this.lfsr[j] = this.lfsr[j+1];  }  lfsr[79] = new\_lfsr;  for (int k = 0; k < 159; k++) {  this.nfsr[k] = this.nfsr[k+1];  }  nfsr[79] = new\_nfsr;  }  } | |

#### 5.1.1.5 Pembuatan Nilai *Keystream*

Alur pembuatan nilai *keystream* pada Grain memiliki alur yang mirip seperti proses inisialisasi kunci, pada pembuatan nilai *keystream* setiap 1 kali clock maka akan menghasilkan satu buah *output* berbeda dengan proses inisialisasi kunci yang dimana hasil dari 1 kali *clock* akan digunakan lagi sebagai *input*an bagi LFSR dan NFSR. Pada proses pembuatan nilai *keystream*, *clocking* dilakukan sebanyak bit dari *plain text* yang akan di enkripsi.

Tabel 5. 9 *Source Code* Pembuatan Nilai *Keystream*

|  |  |
| --- | --- |
| Baris Ke - | *Source Code* |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27 | public String *keystream*(String *input*){  int new\_nfsr = 0;  int new\_lfsr = 0;  int new\_filter;  int *cipher*;  String temp = stringToBinary(*input*);  *System*.out.print("*Keystream* = ");  for (int i = 0; i < temp.length(); i++) {  fx();  gx();  hx();  z();  new\_filter = hx ^ z;  *keystream*[i] = new\_filter;  *System*.out.print(*keystream*[i]);  for (int j = 0; j < 79; j++) {  this.lfsr[j] = this.lfsr[j+1];  }  lfsr[79] = fx;  for (int k = 0; k < 79; k++) {  nfsr[k] = nfsr[k+1];  }  nfsr[79] = gx;  }  *System*.out.println("");  return encrypt(temp);  } |

#### 5.1.1.6 Enkripsi

Pada algoritme grain, ketika sudah mendapatkan nilai *keystream* proses selanjutnya ialah melakukan perhitungan XOR terhadap nilai *keystream* dan nilai *binary* dari *plain text* sehingga menghasilkan *cipher text*.

Tabel 5. 10 *Source Code* Fungsi Enkripsi

|  |  |
| --- | --- |
| Baris Ke - | *Source Code* |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17 | public String encrypt(String *input*){  String result = *input*;  String result\_string = "";  Byte[] result\_array = new Byte[result.length()];  Byte[] result\_xor\_array = new Byte[result.length()];  for (int i = 0; i < result\_array.length; i++) {  result\_array[i] = Byte.parseByte(String.valueOf(result.charAt(i)));  result\_xor\_array[i] = (byte) (this.*keystream*[i] ^ result\_array[i]);  result\_string += result\_xor\_array[i];  binary*keystream* += *keystream*[i];  }  *System*.out.println("*Cipher text* = "+result\_string);  decrypt(binary*keystream*,result\_string);  return result\_string;  } |

#### 5.1.1.7 Dekripsi

Dekripsi adalah proses dimana kita mengembalikan *cipher text* menjadi *plain text* dengan cara melakukan perhitungan XOR nilai *cipher text* terhadap nilai *keystream*.

Tabel 5. 11 *Source Code* Fungsi Dekripsi

|  |  |
| --- | --- |
| Baris Ke - | *Source Code* |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23 | public String decrypt (String *keystream* , String *cipher*){  Byte[] a\_array = new Byte[*cipher*.length()];  Byte[] b\_array = new Byte[*keystream*.length()];  String plain\_binary = "";  String plain = "";  Byte[] hasil = new Byte[*cipher*.length()];  for (int i = 0; i < a\_array.length; i++) {  a\_array[i] =Byte.parseByte(String.valueOf(*cipher*.charAt(i)));  }  for (int i = 0; i < b\_array.length; i++) {  b\_array[i] =Byte.parseByte(String.valueOf(*keystream*.charAt(i)));  }  for (int i = 0; i < *cipher*.length(); i++) {  hasil[i] = (byte) (b\_array[i] ^ a\_array[i]);  plain\_binary += hasil[i];  }  *System*.out.println("Dekripsi = "+plain\_binary);  for (int i = 0; i < plain\_binary.length()-8; i += 8) {  int k = Integer.parseInt(plain\_binary.substring(i,i+8),2);  plain += (char)k;  }  return plain;  } |

### 5.1.2 Implementasi Pada Proses Pemilihan

Pada proses pemilihan, implementasi algoritme grain dan SHA-3 berjalan secara *linear*, dengan maksut bahwa *output* dari algoritme Grain langsung digunakan sebagai *input* - an oleh algoritme SHA-3. Untuk grain memanggil *class grain* yang telah penulis buat pada tahap sebelumnya, sedangkan untuk SHA-3 penulis memanggil *class library* SHA-3 yang telah diunduh sebelumnya. Data yang di enkripsi ialah *string* “paslon1” atau “paslon2” tergantung *radio button* mana yang dipilh oleh *user*. Sedangkan ada beberapa data yang dimasukan kedalam database guna memudahkan penelitian ini antara lain :

1. *Plain text* awal

2. *Cipher text*

3. *Plain text* hasil deskripsi *Cipher text*

4. *Digest text*

Potongan kode programnya adalah sebagai berikut :

Tabel 5. 12 *Source Code* Pemanggilan Fungsi grain dan SHA-3

|  |  |
| --- | --- |
| Baris Ke - | *Source Code* |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27 | public void onClick(View view) {  if (rb\_paslon1.isChecked()) {  grain.init("sekripsi",new*key*);  grain.inisialisasikunci("paslon1");  String encrypt =grain.hasil();  String sha3\_224 = keccak.getHash(encrypt, SHA3\_224);  String id = database*User*.push().get*Key*();  Polling polling = new Polling("paslon1",encrypt,sha3\_224,sha3\_256,sha3\_384,sha3\_512);  database*User*.child(id).setValue(polling);  Toast.makeText(poling*Activity*.this, "Berhasil", Toast.LENGTH\_SHORT).show();  } else if (rb\_paslon2.isChecked()) {  grain.init("sekripsi",new*key*);  grain.inisialisasikunci("paslon2");  String encrypt =grain.hasil();  *System*.out.println(encrypt);  String sha3\_224 = keccak.getHash(encrypt, SHA3\_224);  String id = database*User*.push().get*Key*();  Polling polling = new Polling("paslon1",encrypt,sha3\_224,sha3\_256,sha3\_384,sha3\_512);  database*User*.child(id).setValue(polling);  Toast.makeText(poling*Activity*.this, "Berhasil", Toast.LENGTH\_SHORT).show();  } }  }); |

## 5.2 Parameter Pengujian

Pada penelitian ini menggunakan beberapa parameter pengujian untuk menguji algoritme grain dan algoritme SHA-3.

1. Pengujian validitas *test vector*

2. Waktu yang digunakan oleh algoritme grain dalam menghasilkan *cipher text*

3. Pengujian enkripsi dan dekripsi pada kedua algoritme

4. Waktu total yang diperlukan algoritme grain dan SHA-3 dalam memproses *plain text* hingga menjadi *digest text*.

## 5.3 *Test Vector*

### 5.3.1 Tujuan Pengujian

Tujuan pengujian *test vector* ialah memastikan bahwa algoritme Grain yang telah dibuat oleh penulis memiliki *output* *keystream* sama seperti dengan algoritme grain yang telah dibuat oleh penciptanya. Begitu juga dengan algoritme SHA-3, agar code dari library yang digunakan oleh penulis pada penelitian ini memiliki *digest text* yang sama dengan algoritme SHA-3 yang telah dibuat oleh penciptanya. Sehingga code yang digunakan pada penelitian ini bersifat valid.

### 5.3.2 Prosedur Pengujian

Prosedur pengujian *test vector* dilakukan dengan cara memberikan masukan yang telah ditentukan pada *test vector* masing-masing algoritm lalu mencocokan *output* yang dihasilkan, pada algoritme grain mencocokan nilai *keystream* dan pada algoritme SHA-3 mencocokan nilai *digest text*. *Test vector* yang digunakan pada pengujian ini merujuk pada paper berjudul Grain - *A Stream Cipher for Constrained Environments* untuk algoritme Grain dan untuk SHA-3 merujuk pada situs *http://www.di-mgt.com.au/sha\_testvectors.htm*. *Test vector* algoritme Grain, format yang digunakan ialah hexadecimal baik untuk *IV*, *key* maupun *keystream*nya. Untuk SHA-3 format yang digunakan adalah string sebagai masukan.

### 5.3.3 Hasil dan Analisis

#### 5.3.3.1 Algoritme Grain

*Test vector* untuk algoritme Grain.

1. *Key* : 00000000000000000000

*IV* : 0000000000000000

*Keystream* : 7b978cf36846e5f4ee0b

1. *Key* : 0123456789abcdef1234

*IV* : 0123456789abcdef

*Keystream* : 42b567ccc65317680225

Hasil *keystream* dari algoritme Grain yang digunakan pada penelitian ini

1. *Key* : 00000000000000000000

*IV* : 0000000000000000

*Keystream* : 7b978cf36846e5f4ee0b

1. *Key* : 0123456789abcdef1234

*IV* : 0123456789abcdef

*Keystream* : 42b567ccc65317680225

Dari kedua hasil percobaan diatas membuktikan bahwa hasil *keystream* yang digunakan oleh penulis dalam penelitian ini memiliki hasil *keystream* yang sama pada *test vector* yang telah disediakan pada algoritme grain, sehingga *code* yang ditulis oleh penulis bersifat *valid.*

#### 5.3.3.2 Algoritme SHA-3

*Test vector* untuk algoritme SHA-3

1. Pesan: “abc”

SHA-3: e642824c3f8cf24a d09234ee7d3c766f c9a3a5168d0c94ad 73b46fdf

1. Pesan: “” (*string* kosong)

SHA-3: 6b4e03423667dbb7 3b6e15454f0eb1ab d4597f9a1b078e3f 5b5a6bc7

1. Pesan: "abcdbcdecdefdefgefghfghighijhijkijkljklmklmnlmnomnopnopq"

SHA-3: 8a24108b154ada21 c9fd5574494479ba 5c7e7ab76ef264ea d0fcce33

Hasil hash dari algoritme SHA-3 yang digunakan pada penelitian ini

1. Pesan: “abc”

SHA-3: e642824c3f8cf24a d09234ee7d3c766f c9a3a5168d0c94ad 73b46fdf

1. Pesan: “” (*string* kosong)

SHA-3: 6b4e03423667dbb7 3b6e15454f0eb1ab d4597f9a1b078e3f 5b5a6bc7

1. Pesan: "abcdbcdecdefdefgefghfghighijhijkijkljklmklmnlmnomnopnopq"

SHA-3: 8a24108b154ada21 c9fd5574494479ba 5c7e7ab76ef264ea d0fcce33

Dari ketiga hasil percobaan diatas membuktikan bahwa hasil *digest text* yang digunakan oleh penulis dalam penelitian ini memiliki hasil yang sama pada *test vector* yang mengacu pada situs *https://www.di-mgt.com.au/sha\_testvectors.html ,* sehingga *library code* yang digunakan oleh penulis bersifat valid.

## 5.4 Pengujian Pemrosesan *Cipher text*

### 5.4.1 Tujuan Pengujian

Tujuan pengujian ini ialah untuk mengetahui dan membandingkan waktu yang ditempuh aplikasi pemilu *online* berbasis *android* ketika belum diberi fungsi enkripsi algoritme grain dan setelah diberi fungsi enkripsi algoritme grain untuk menghasilkan *cipher* **text** dengan memanfaatkan inputan berupa *IV,* *key* dan juga *plain text* .

### 5.4.2 Prosedur Pengujian

Untuk melakuan pengujian pemrosesan *cipher text*  pada aplikasi pemilu online berbasis android, diperlukan sebuah akun untuk dapat masuk ke dalam sistem dan melakukan pengujian . Pada pengujian ini akan digunakan sebuah akun dengan NIK = “1451502071110510” dan *password* “arsana” . Dari NIK dan *password* tersebut akan tercipta sebuah nilai baru sebagai nilai key yang digunakan algoritme grain untuk menghasilkan keystream . Mekanisme pembentukan nilai keynya ialah mengambil 6 *digit* awal *password* lalu ditambahkan dengan 4 *digit* terakhir dari NIK agar sesuai dengan kriteria key dari algoritme grain yaitu maksimal sepanjang 10 karakter atau sepanjang 80 bit , sehingga pada pengujian ini menghasilkan nilai key yaitu “arsana0510” . Untuk nilai IV diatur secara default yaitu bernilai “sekripsi” sesuai dengan kriteria IV dari algoritme grain yaitu maksimal sepanjang 8 karakter atau 64 bit . Plain text yang digunakan juga diatur secara default yaitu “paslon1” . Waktu pengujian akan dimulai ketika tombol “Pilih” pada halaman pemilihan ditekan dan berhenti ketika data telah dikirimkan menuju database . Pengujian ini akan dilakukan berulang-ulang selama 30 kali dengan *key* , *IV* dan *plain text* yang sama. Pengujian ini akan dilakukan berulang-ulang selama 30 kali dengan skenario yang sama .

### 5.4.3 Hasil Pengujian

Berikut merupakan waktu tempuh yang dihasilkan oleh aplikasi pemilu online berbasis android baik yang tidak menerapkan fungsi algoritme grain maupun yang menerapkan fungsi algoritme grain .

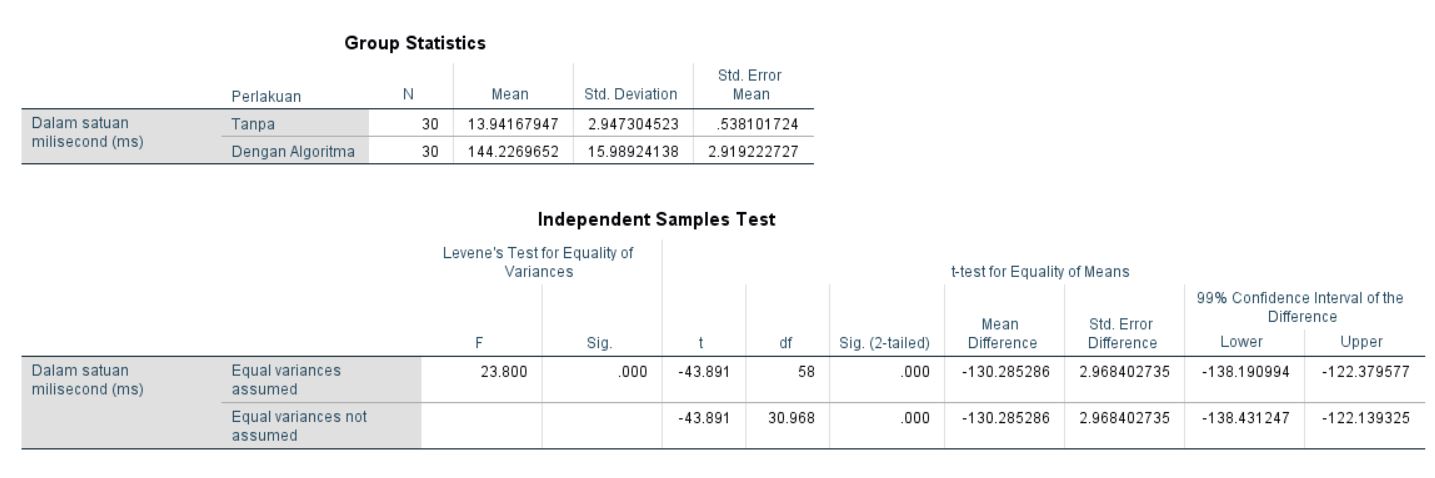
Tabel 5. 13 Hasil pengujian waktu tempuh aplikasi pemilu online berbasis android dengan grain maupun tanpa grain

|  |  |  |  |
| --- | --- | --- | --- |
| Percobaan Ke- | Waktu Tempuh tanpa Algoritme Grain (ms) | Waktu Tempuh dengan Algoritme Grain (ms) | Selisih Waktu Tempuh (ms) |
| 1. | 21,537461 | 169,162385 | 147,624924 |
| 2. | 12,225 | 165,383731 | 153,158731 |
| 3. | 12,0135 | 139,472693 | 127,459193 |
| 4. | 13,671 | 102,404731 | 88,733731 |
| 5. | 14,331923 | 142,04323 | 127,711307 |
| 6. | 15,988807 | 138,9125 | 122,923693 |
| 7. | 16,251577 | 160,197461 | 143,945884 |
| 8. | 14,523231 | 143,584538 | 129,061307 |
| 9. | 11,890731 | 154,275 | 142,384269 |
| 10. | 13,143654 | 140,913423 | 127,769769 |
| 11. | 8,334039 | 143,464769 | 135,13073 |
| 12. | 8,148654 | 138,180269 | 130,031615 |
| 13. | 15,478461 | 109,977846 | 94,499385 |
| 14. | 13,907192 | 149,198116 | 135,290924 |
| 15. | 13,836577 | 142,218884 | 128,382307 |
| 16. | 11,8875 | 129,131538 | 117,244038 |
| 17. | 18,197808 | 146,146307 | 127,948499 |
| 18. | 16,829192 | 122,489423 | 105,660231 |
| 19. | 18,400538 | 112,027038 | 93,6265 |
| 20. | 15,565577 | 157,748576 | 142,182999 |
| 21. | 14,872615 | 149,820962 | 134,948347 |
| 22. | 16,187231 | 154,621346 | 138,434115 |
| 23. | 10,834308 | 141,776346 | 130,942038 |
| 24. | 12,788077 | 158,588846 | 145,800769 |
| 25. | 13,691192 | 157,380384 | 143,689192 |
| 26. | 11,91 | 151,664846 | 139,754846 |
| 27. | 12,642923 | 159,303038 | 146,660115 |
| 28. | 12,642923 | 155,828346 | 143,185423 |
| 29. | 16,910193 | 151,150577 | 134,240384 |
| 30. | 9,6085 | 139,741808 | 130,133308 |
| Rata-Rata | 13,94167947 | 144,226965 | 130,285286 |

Tabel 5.2 menjelaskan tentang waktu tempuh yang dihasilkan oleh aplikasi pemilu online berbasis andoid ketika menerapkan algoritme grain maupun tidak yang diulangi sebanyak 30 kali. Dari tabel tersebut juga menunjukan waktu rata-rata yang dihasilkan oleh aplikasi pemilu online berbasis android ketika digunakan . Dari hasil pengujian tersebut, berikut gambar 5.1 yang merupakan grafik dari hasil pengujian :

Gambar 5. 1 Grafik Hasil Waktu Tempuh Aplikasi Pemilu Online Berbasis Android dengan Grain Maupun tanpa Grain.

Untuk melihat perbedaan waktu tempuh aplikasi pemilu *online* berbasis *android* dengan grain maupun tanpa grain maka dilakukanlah *independent sampe t-test* pada data hasil pengujian tersebut . Hasil pengujian *independent sample t-test* perbedaan waktu tempuh aplikasi pemilu *online* berbasis *android* dengan grain maupun tanpa grain dapa dilihat pada Tabel 5.3

Tabel 5. 14 *Independent samples t-Test* untuk pengujian waktu tempuh aplikasi pemilu online berbasis android dengan grain maupun tanpa grain 

Ketentuan pengambilan keputusan didasarkan pada ketentuan sebagai berikut:

Hipotesis:

H0: Tidak ada perbedaan yang signifikan antara waktu tempuh aplikasi pemilu online berbasis android ketika menggunakan grain dengan tidak menggunakan grain.

H1: Terdapat perbedaan yang signifikan antara waktu tempuh aplikasi pemilu online berbasis android ketika menggunakan grain dengan tidak menggunakan grain.

Kriteria keputusan:

1. Jika probabilitas (Sig.) > 0,05 maka H0 diterima
2. Jika probabilitas (Sig.) < 0,05 maka H0 ditolak

Diketahui nilai t dari hasil perhitungan t-test dari hasil pengujian waktu tempuh aplikasi pemilu online berbasis android dengan grain maupun tanpa grain adalah -43,891 dengan probabilitas (Sig.) 0,000. Karena probabilitas 0,000 < 0,05 maka H0 ditolak. Artinya terdapat perbedaan yang signifikan antara waktu tempuh aplikasi pemilu online berbasis android dengan grain maupun tanpa grain. Berdasarkan hasil pengolahan data tersebut, maka dapat disimpulkan bahwa hasil pengujian waktu tempuh aplikasi pemilu online berbasis android dengan grain maupun tanpa grain memiliki perbedaan yang signifikan. Dalam penelitian ini aplikasi pemilu online berbasis android yang menerapkan algoritme grain memiliki waktu pemrosesan yang lebih lama dibandingkan dengan yang tidak menerapkan algoritme grain.

## 5.5 Pengujian Pemrosesan *Digest text*

### 5.5.1 Tujuan Pengujian

Tujuan pengujian ini ialah untuk mengetahui dan membandingkan waktu yang ditempuh aplikasi pemilu *online* berbasis android ketika belum diberi fungsi *hash* algoritme SHA-3 dan setelah diberi fungsi *hash* algoritme SHA-3 untuk menghasilkan *digest text* .

### 5.5.2 Prosedur Pengujian

Untuk melakuan pengujian pemrosesan *digest text* pada aplikasi pemilu *online* berbasis android, diperlukan sebuah akun untuk dapat masuk ke dalam sistem dan melakukan pengujian . Pada pengujian ini akan digunakan sebuah akun dengan NIK = “1451502071110510” dan *password* “arsana” . *Plain text* yang digunakan diatur secara *default* yaitu “paslon1” . Waktu pengujian akan dimulai ketika tombol “Pilih” pada aplikasi ditekan dan berhenti ketika data dikirim menuju database . Pengujian ini akan dilakukan berulang-ulang selama 30 kali dengan *plain text* yang sama.

### 5.5.3 Hasil Pengujian

Berikut merupakan waktu tempuh yang dihasilkan oleh algoritme SHA-3 dalam menghasilkan *digest text* sesuai dengan prosedur yang telah dijelaskan pada sub-bab sebelumnya.

Tabel 5. 15 Hasil pengujian waktu tempuh aplikasi pemilu online berbasis android dengan SHA-3 maupun tanpa SHA-3

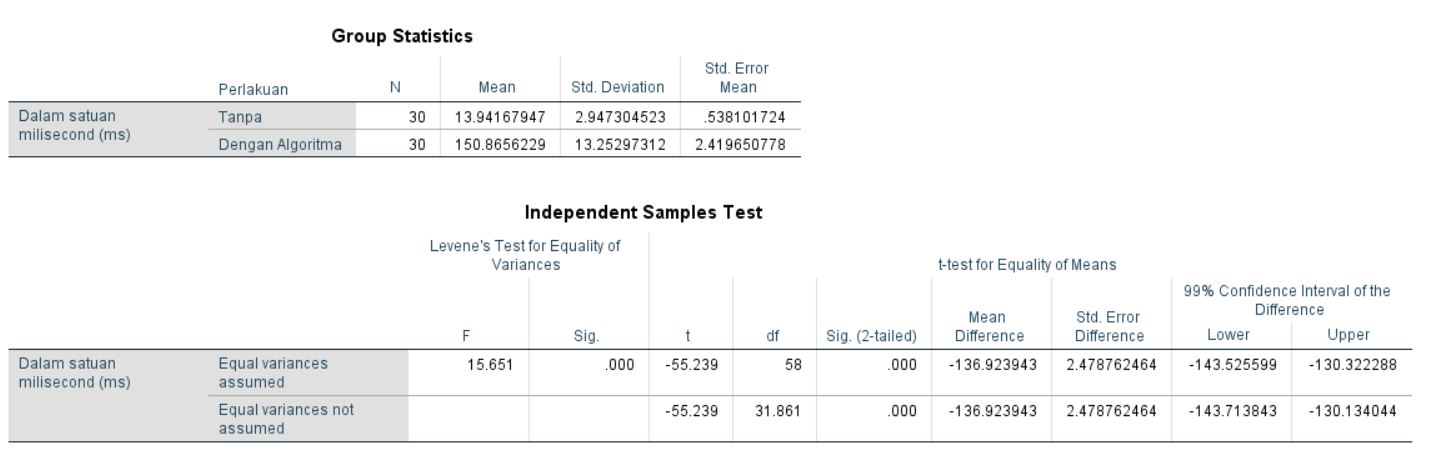
|  |  |  |  |
| --- | --- | --- | --- |
| Percobaan Ke- | Waktu Tempuh tanpa algoritme SHA-3 (ms) | Waktu Tempuh dengan Algoritme SHA-3 (ms) | Selisih Waktu Tempuh (ms) |
| 1. | 21,537461 | 156,452 | 134,9145 |
| 2. | 12,225 | 159,3475 | 147,1225 |
| 3. | 12,0135 | 150,2624 | 138,2489 |
| 4. | 13,671 | 151,6048 | 137,9338 |
| 5. | 14,331923 | 117,2692 | 102,9373 |
| 6. | 15,988807 | 159,0314 | 143,0426 |
| 7. | 16,251577 | 183,7567 | 167,5052 |
| 8. | 14,523231 | 151,6672 | 137,1439 |
| 9. | 11,890731 | 154,6929 | 142,8022 |
| 10. | 13,143654 | 154,1173 | 140,9736 |
| 11. | 8,334039 | 131,0592 | 122,7251 |
| 12. | 8,148654 | 152,1585 | 144,0099 |
| 13. | 15,478461 | 134,5463 | 119,0678 |
| 14. | 13,907192 | 136,2127 | 122,3055 |
| 15. | 13,836577 | 149,9727 | 136,1362 |
| 16. | 11,8875 | 160,6645 | 148,777 |
| 17. | 18,197808 | 167,7928 | 149,595 |
| 18. | 16,829192 | 155,6505 | 138,8213 |
| 19. | 18,400538 | 145,3677 | 126,9672 |
| 20. | 15,565577 | 149,7287 | 134,1631 |
| 21. | 14,872615 | 122,8543 | 107,9817 |
| 22. | 16,187231 | 160,3451 | 144,1579 |
| 23. | 10,834308 | 150,5376 | 139,7033 |
| 24. | 12,788077 | 135,4426 | 122,6545 |
| 25. | 13,691192 | 149,4408 | 135,7497 |
| 26. | 11,91 | 155,2312 | 143,3212 |
| 27. | 12,642923 | 163,0055 | 150,3625 |
| 28. | 12,642923 | 157,4566 | 144,8137 |
| 29. | 16,910193 | 152,6402 | 135,73 |
| 30. | 9,6085 | 157,6598 | 148,0513 |
| Rata-Rata | 13,94167947 | 150,8656 | 136,9239 |

Tabel 5.4 menjelaskan tentang waktu tempuh yang dihasilkan oleh aplikasi pemilu online berbasis andoid ketika menerapkan algoritme SHA-3 maupun tidak yang diulangi sebanyak 30 kali. Dari tabel tersebut juga menunjukan waktu rata-rata yang dihasilkan oleh aplikasi pemilu online berbasis android ketika digunakan . Dari hasil pengujian tersebut, berikut gambar 5.2 yang merupakan grafik dari hasil pengujian :

Gambar 5. 2 Grafik Hasil Waktu Tempuh Aplikasi Pemilu *Online* Berbasis Android dengan SHA-3 Maupun tanpa SHA-3

Untuk melihat perbedaan waktu tempuh aplikasi pemilu online berbasis android dengan SHA-3 maupun tanpa SHA-3 maka dilakukanlah *independent sampe t-test* pada data hasil pengujian tersebut . Hasil pengujian *independent sample t-test* perbedaan waktu tempuh aplikasi pemilu online berbasis android dengan SHA-3 maupun tanpa SHA-3 dapa dilihat pada Tabel 5.5

Tabel 5. 16 *Independent samples t-Test* untuk pengujian waktu tempuh aplikasi pemilu *online* berbasis android dengan SHA-3 maupun tanpa SHA-3



Ketentuan pengambilan keputusan didasarkan pada ketentuan sebagai berikut:

Hipotesis:

H0: Tidak ada perbedaan yang signifikan antara waktu tempuh aplikasi pemilu online berbasis android ketika menggunakan SHA-3 dengan tidak menggunakan SHA-3.

H1: Terdapat perbedaan yang signifikan antara waktu tempuh aplikasi pemilu online berbasis android ketika menggunakan SHA-3 dengan tidak menggunakan SHA-3.

Kriteria keputusan:

1. Jika probabilitas (Sig.) > 0,05 maka H0 diterima.
2. Jika probabilitas (Sig.) < 0,05 maka H0 ditolak.

Diketahui nilai t dari hasil perhitungan t-test dari hasil pengujian waktu tempuh aplikasi pemilu online berbasis android dengan grain maupun tanpa grain adalah -55,239 dengan probabilitas (Sig.) 0,000. Karena probabilitas 0,000 < 0,05 maka H0 ditolak. Artinya terdapat perbedaan yang signifikan antara waktu tempuh aplikasi pemilu online berbasis android dengan SHA-3 maupun tanpa SHA-3. Berdasarkan hasil pengolahan data tersebut, maka dapat disimpulkan bahwa hasil pengujian waktu tempuh aplikasi pemilu online berbasis android dengan SHA-3 maupun tanpa SHA-3 memiliki perbedaan yang signifikan. Dalam penelitian ini aplikasi pemilu online berbasis android yang menerapkan algoritme SHA-3 memiliki waktu pemrosesan yang lebih lama dibandingkan dengan yang tidak menerapkan algoritme SHA-3.

## 5.6 Pengujian Pemrosesan Seluruh Sistem

### 5.6.1 Tujuan Pengujian

Tujuan pengujian ini ialah untuk mengetahui dan membandingkan waktu yang ditempuh aplikasi pemilu *online* berbasis android ketika belum diterapkan algoritme grain berserta SHA-3 dengan setelah diterapkan algoritme grain berserta SHA-3 sebagai fungsi untuk enkripsi dan *hash* data.

### 5.6.2 Prosedur Pengujian

Untuk melakuan pengujian pemrosesanseluruh sistem pada aplikasi pemilu *online* berbasis android, diperlukan sebuah akun untuk dapat masuk ke dalam sistem dan melakukan pengujian . Pada pengujian ini akan digunakan sebuah akun dengan NIK = “1451502071110510” dan *password* “arsana” . Dari NIK dan *password* tersebut akan tercipta sebuah nilai baru sebagai nilai *key* yang digunakan algoritme grain untuk menghasilkan keystream . Mekanisme pembentukan nilai *key*nya ialah mengambil 6 *digit* awal *password* lalu ditambahkan dengan 4 *digit* terakhir dari NIK agar sesuai dengan kriteria *key* dari algoritme grain yaitu maksimal sepanjang 10 karakter atau sepanjang 80 *bit* , sehingga pada pengujian ini menghasilkan nilai key yaitu “arsana0510” . Untuk nilai IV diatur secara default yaitu bernilai “sekripsi” sesuai dengan kriteria IV dari algoritme grain yaitu maksimal sepanjang 8 karakter atau 64 bit . Plain text yang digunakan juga diatur secara default yaitu “paslon1” . *Output* yang dikeluarkan oleh algoritme grain akan dijadikan *input* oleh algoritme SHA-3 untuk menghasilkan *digest text* sepanjang 224 bit . Waktu pengujian akan dimulai ketika tombol “Pilih” pada halaman pemilihan ditekan dan berhenti ketika aplikasi telah mengirimkan data menuju database . Pengujian ini akan dilakukan berulang-ulang selama 30 kali dengan *key* , *IV* dan *plain text* yang sama. Pengujian ini akan dilakukan berulang-ulang selama 30 kali dengan skenario yang sama .

### 5.6.3 Hasil Pengujian

Berikut merupakan waktu tempuh yang dihasilkan oleh aplikasi pemilu *online* berbasis android jika dibandingkan sebelum dan sesudah diterapkanya algoritme grain dan SHA-3.

Tabel 5. 17 Hasil pengujian waktu pemrosesan seluruh sistem

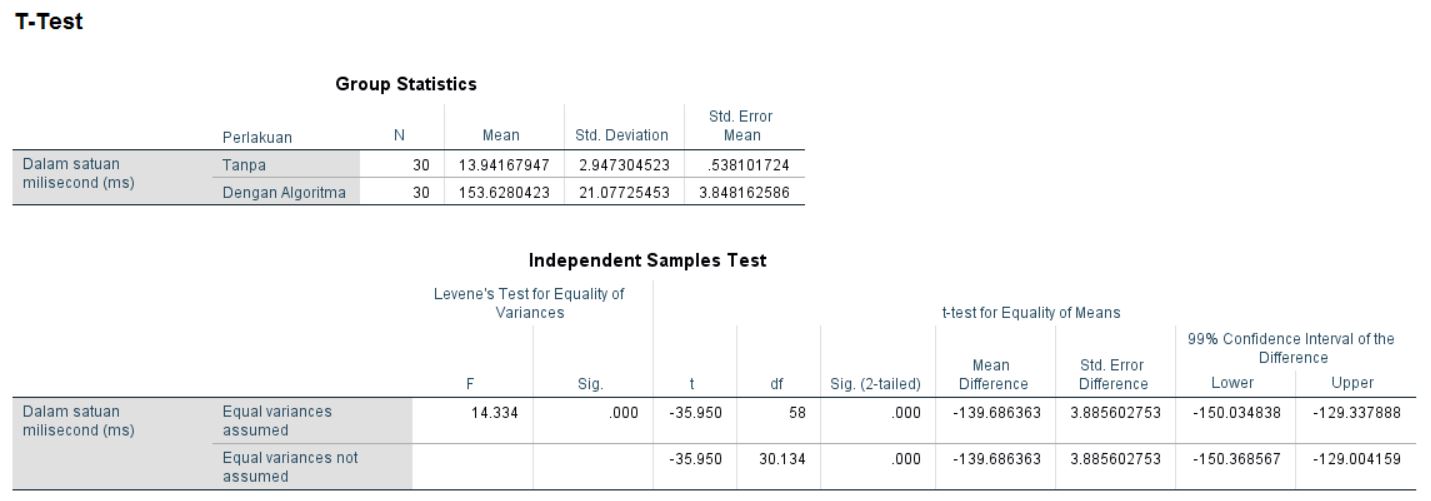
|  |  |  |  |
| --- | --- | --- | --- |
| Percobaan Ke- | Waktu Tempuh tanpa Algoritme Grain dan SHA-3 (ms) | Waktu Tempuh dengan Algoritme Grain dan SHA-3 (ms) | Selisih Waktu Tempuh (ms) |
| 1. | 21,537461 | 174,503115 | 152,97 |
| 2. | 12,225 | 176,3375 | 164,11 |
| 3. | 12,0135 | 147,04723 | 135,03 |
| 4. | 13,671 | 142,554462 | 128,88 |
| 5. | 14,331923 | 151,047076 | 136,72 |
| 6. | 15,988807 | 158,281077 | 142,29 |
| 7. | 16,251577 | 167,617307 | 151,37 |
| 8. | 14,523231 | 142,343231 | 127,82 |
| 9. | 11,890731 | 147,048269 | 135,16 |
| 10. | 13,143654 | 143,142808 | 130 |
| 11. | 8,334039 | 244,339846 | 236,01 |
| 12. | 8,148654 | 139,659462 | 131,51 |
| 13. | 15,478461 | 154,715385 | 139,24 |
| 14. | 13,907192 | 128,254192 | 114,35 |
| 15. | 13,836577 | 139,049808 | 125,21 |
| 16. | 11,8875 | 140,095039 | 128,21 |
| 17. | 18,197808 | 144,497384 | 126,3 |
| 18. | 16,829192 | 149,742923 | 132,91 |
| 19. | 18,400538 | 148,580423 | 130,18 |
| 20. | 15,565577 | 150,952769 | 135,39 |
| 21. | 14,872615 | 160,856654 | 145,98 |
| 22. | 16,187231 | 144,904423 | 128,72 |
| 23. | 10,834308 | 168,285846 | 157,45 |
| 24. | 12,788077 | 155,860077 | 143,07 |
| 25. | 13,691192 | 135,702807 | 122,01 |
| 26. | 11,91 | 156,489 | 144,58 |
| 27. | 12,642923 | 169,259693 | 156,62 |
| 28. | 12,642923 | 134,721962 | 122,08 |
| 29. | 16,910193 | 158,390846 | 141,48 |
| 30. | 9,6085 | 134,560654 | 124,95 |
| Rata-Rata | 13,94167947 | 153,6280423 | 139,69 |

Tabel 5.6 menjelaskan tentang waktu tempuh yang dihasilkan oleh aplikasi pemilu *online* berbasis andoid ketika menerapkan algoritme grain serta SHA-3 maupun tidak yang diulangi sebanyak 30 kali. Dari tabel tersebut juga menunjukan waktu rata-rata yang dihasilkan oleh aplikasi pemilu *online* berbasis android ketika digunakan . Dari hasil pengujian tersebut, berikut gambar 5.3 yang merupakan grafik dari hasil pengujian :

Gambar 5. 3 Grafik Hasil Waktu Tempuh Aplikasi Pemilu Online Berbasis Android dengan Grain dan SHA-3 maupun tanpa Grain dan SHA-3

Untuk melihat perbedaan waktu tempuh aplikasi pemilu online berbasis android dengan SHA-3 maupun tanpa SHA-3 maka dilakukanlah *independent sampe t-test* pada data hasil pengujian tersebut . Hasil pengujian *independent sample t-test* perbedaan waktu tempuh aplikasi pemilu online berbasis android dengan SHA-3 maupun tanpa SHA-3 dapa dilihat pada Tabel 5.7

Tabel 5. 18 *Independent samples t-Test* untuk pengujian waktu tempuh aplikasi pemilu online berbasis android dengan Grain dan SHA-3 maupun tanpa grain dan SHA-3



Ketentuan pengambilan keputusan didasarkan pada ketentuan sebagai berikut:

Hipotesis:

H0: Tidak ada perbedaan yang signifikan antara waktu tempuh aplikasi pemilu online berbasis android ketika menggunakan grain dan SHA-3 dengan tidak menggunakan grain dan SHA-3.

H1: Terdapat perbedaan yang signifikan antara waktu tempuh aplikasi pemilu online berbasis android ketika menggunakan grain dan SHA-3 dengan tidak menggunakan grain dan SHA-3.

Kriteria keputusan:

1. Jika probabilitas (Sig.) > 0,05 maka H0 diterima
2. Jika probabilitas (Sig.) < 0,05 maka H0 ditolak

Diketahui nilai t dari hasil perhitungan t-test dari hasil pengujian waktu tempuh aplikasi pemilu online berbasis android dengan grain dan SHA-3 maupun tanpa grain dan SHA-3 adalah -35,950 dengan probabilitas (Sig.) 0,000. Karena probabilitas 0,000 < 0,05 maka H0 ditolak. Artinya terdapat perbedaan yang signifikan antara waktu tempuh aplikasi pemilu online berbasis android dengan grain dan SHA-3 maupun tanpa grain dan SHA-3 . Berdasarkan hasil pengolahan data tersebut, maka dapat disimpulkan bahwa hasil pengujian waktu tempuh aplikasi pemilu online berbasis android dengan grain dan SHA-3 maupun tanpa grain dan SHA-3 memiliki perbedaan yang signifikan. Dalam penelitian ini aplikasi pemilu online berbasis android yang menerapkan algoritme grain dan SHA-3 memiliki waktu pemrosesan yang lebih lama dibandingkan dengan yang tidak menerapkan algoritme grain.

## 5.7 Pengujian *Confidentiality*

### 5.7.1 Tujuan Pengujian

Pengujian *Confidentiality* dilakukan untuk memastikan bahwa data dienkripsi dengan benar sesuai dengan algoritme yang digunakan . Untuk memastikan ketepatanya , maka perlu dicocokan nilai dari *plain text* pada saat sebelum di proses dengan *plain text* hasil dari proses dekripsi.

### 5.7.2 Prosedur Pengujian

Prosedur pengujian yang digunakan pada pengujian ini adalah dengan cara mencocokan nilai *plain text* yang di*input*kan dengan *plain text* yang dihasilkan oleh method decrypt . Jika hasilnya cocok maka code yang digunakan dalam penelitian ini bersifat valid . Nilai *key* dan *IV* diberi nilai default masing masing ialah “sekripsi” dan “arsana10” , untuk *plain text* akan menggunakan 5 buah *plain text* yang berbeda.

### 5.7.3 Hasil Pengujian

Pada hasil pengujian ini segala hasil yang ditampilkan sudah dirubah kedalam bentuk binary .

*Key* = “arsana0510”

*IV* = “sekripsi”

Plaint text *input* = “paslon1”

Tabel 5. 19 Tabel pengujian *confidentiality* pertama

|  |  |  |  |
| --- | --- | --- | --- |
| *Key* | *IV* | *Plain text* *Input* | *Plain text* *Output* |
| 0110000101110010011100110110000101101110011000010011000100110000 | 0111001101100101011010110111001001101001011100000111001101101001 | 01110000011000010111001101101100011011110110111000110001 | 01110000011000010111001101101100011011110110111000110001 |

*Key* = “arsana0510”

*IV* = “sekripsi”

Plaint text *input* = “paslon2”

Tabel 5. 20 Tabel pengujian *confidentiality* kedua

|  |  |  |  |
| --- | --- | --- | --- |
| *Key* | *IV* | *Plain text* *Input* | *Plain text* *Output* |
| 0110000101110010011100110110000101101110011000010011000100110000 | 0111001101100101011010110111001001101001011100000111001101101001 | 01110000011000010111001101101100011011110110111000110010 | 01110000011000010111001101101100011011110110111000110010 |

*Key* = “arsana0510”

*IV* = “sekripsi”

Plaint text *input* = “ ” (spasi)

Tabel 5. 21 Tabel pengujian *confidentiality* ketiga

|  |  |  |  |
| --- | --- | --- | --- |
| *Key* | *IV* | *Plain text* *Input* | *Plain text* *Output* |
| 0110000101110010011100110110000101101110011000010011000100110000 | 0111001101100101011010110111001001101001011100000111001101101001 | 00100000 | 00100000 |

*Key* = “arsana0510”

*IV* = “sekripsi”

Plaint text *input* = “abcdef12345”

Tabel 5. 22 Tabel pengujian *confidentiality* keempat

|  |  |  |  |
| --- | --- | --- | --- |
| *Key* | *IV* | *Plain text* *Input* | *Plain text* *Output* |
| 0110000101110010011100110110000101101110011000010011000100110000 | 0111001101100101011010110111001001101001011100000111001101101001 | 0110000101100010011000110110010001100101011001100011000100110010001100110011010000110101 | 0110000101100010011000110110010001100101011001100011000100110010001100110011010000110101 |

*Key* = “arsana0510”

*IV* = “sekripsi”

Plaint text *input* = “!@#$%^&\*()”

Tabel 5. 23 Tabel pengujian *confidentiality* kelima

|  |  |  |  |
| --- | --- | --- | --- |
| *Key* | *IV* | *Plain text* *Input* | *Plain text* *Output* |
| 0110000101110010011100110110000101101110011000010011000100110000 | 0111001101100101011010110111001001101001011100000111001101101001 | 00100001010000000010001100100100001001010101111000100110001010100010100000101001 | 00100001010000000010001100100100001001010101111000100110001010100010100000101001 |

## 5.8 Pengujian *Integrity*

### 5.8.1 Tujuan Pengujian

Pengujian *Integrity* dilakukan untuk memastikan bahwa data di *hashing* dengan benar sesuai dengan algoritme yang digunakan . Untuk memastikan ketepatanya , maka perlu dilakukan pengujian *integrity* dengan cara mencocokan hasil *digest text* . Berbeda dengan algoritme enkripsi yang mengizinkan *cipher text* nya untuk di dekripsi , algoritme hash tidak mengizinkan *digest text* untuk dikembalikan ke bentuk semula (*plain text*) . Maka satu satunya cara untuk mencocokanya adalah dengan cara melakukan minimal dua kali percobaan dengan *input* yang sama , lalu jika *output* dari kedua percobaan tersebut bernilai sama maka algoritme yang digunakan bersifat valid .

### 5.8.2 Prosedur Pengujian

Prosedur pengujian yang digunakan pada pengujian ini adalah dengan cara mencocokan nilai *digest text* percobaan pertama dengan nilai *digest text* pada percobaan kedua. Jika hasilnya cocok maka code yang digunakan dalam penelitian ini bersifat valid . Untuk nilai *input* yang digunakan pada pengujian ini memanfaatkan *output* dari algoritme grain yang berupa *cipher text* . Oleh karena itu kita juga harus mengatur nilai *key* dan *IV* pada algoritme grain. *Key* dan *IV* diberi nilai default masing masing ialah “sekripsi” dan “arsana10” , untuk *plain text* akan menggunakan 5 buah *plain text* yang berbeda.

### 5.8.3 Hasil Pengujian

Pada hasil pengujian ini segala hasil yang ditampilkan sudah dirubah kedalam bentuk binary .

*Key* = “arsana0510”

*IV* = “sekripsi”

Plaint text *input* = “paslon1”

Tabel 5. 24 Tabel pengujian *integrity* pertama

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| *Key* | *IV* | *Cipher text* Hasil  Grain | *Digest text*  Percobaan  Ke-1 | *Digest text* Percobaan Ke-2 |
| 0110000101110010011100110110000101101110011000010011000100110000 | 0111001101100101011010110111001001101001011100000111001101101001 | 01010111000010010011110000110110011010010011100111011101 | d927b8753b810d72042ff278aa4628debf20c9326bbf34c7eda4be5a | d927b8753b810d72042ff278aa4628debf20c9326bbf34c7eda4be5a |

*Key* = “arsana0510”

*IV* = “sekripsi”

Plaint text *input* = “paslon2”

Tabel 5. 25 Tabel pengujian *integrity* kedua

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| *Key* | *IV* | *Cipher text* Hasil  Grain | *Digest text*  Percobaan  Ke-1 | *Digest text* Percobaan Ke-2 |
| 0110000101110010011100110110000101101110011000010011000100110000 | 0111001101100101011010110111001001101001011100000111001101101001 | 01010111000010010011110000110110011010010011100111011110 | e47dc6af54012a1155c847d4a3a2eae80dc82a628544ba884456927f | e47dc6af54012a1155c847d4a3a2eae80dc82a628544ba884456927f |

*Key* = “arsana0510”

*IV* = “sekripsi”

Plaint text *input* = “ ” (spasi)

Tabel 5. 26 Tabel pengujian *integrity* ketiga

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| *Key* | *IV* | *Cipher text* Hasil Grain | *Digest text*  Percobaan  Ke-1 | *Digest text* Percobaan Ke-2 |
| 0110000101110010011100110110000101101110011000010011000100110000 | 0111001101100101011010110111001001101001011100000111001101101001 | 00000111 | fe91600c40fbb11e979a32c59ed8fef1357caef9a3b333b3dd4ece3a | fe91600c40fbb11e979a32c59ed8fef1357caef9a3b333b3dd4ece3a |

*Key* = “arsana0510”

*IV* = “sekripsi”

Plaint text *input* = “abcdef12345”

Tabel 5. 27 Tabel pengujian *integrity* keempat

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| *Key* | *IV* | *Cipher text* Hasil Grain | *Digest text*  Percobaan  Ke-1 | *Digest text*  Percobaan Ke-2 |
| 0110000101110010011100110110000101101110011000010011000100110000 | 0111001101100101011010110111001001101001011100000111001101101001 | 0100011000001010001011000011111001100011001100011101110100001011011011000110000011100101 | da55012e0a59281bd2955a66be11868ff0986ea4e466a9f730c5aac1 | da55012e0a59281bd2955a66be11868ff0986ea4e466a9f730c5aac1 |

*Key* = “arsana0510”

*IV* = “sekripsi”

Plaint text *input* = “!@#$%^&\*()”

Tabel 5. 28 Tabel pengujian *integrity* kelima

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| *Key* | *IV* | *Cipher text* Hasil Grain | *Digest text*  Percobaan  Ke-1 | *Digest text*  Percobaan Ke-2 |
| 0110000101110010011100110110000101101110011000010011000100110000 | 0111001101100101011010110111001001101001011100000111001101101001 | 00000110001010000110110001111110001000110000100111001010000100110111011101111101 | 4dd4c79a18aade0295e8b942cf44b31298f5b440ee11ea2a3b5c207a | 4dd4c79a18aade0295e8b942cf44b31298f5b440ee11ea2a3b5c207a |

## 5.9 Pengujian Skema Pembentukan Key

### 5.9.1 Tujuan Pengujian

Pada aplikasi pemilu online berbasis android yang telah penulis buat , menerapkan suatu skema pembuatan key untuk inputan algoritme grain dengan memanfaatkan kombinasi dari NIK dan *Password* dari *user* untuk memberikan tingkat keamanan yang lebih kepada *user* . Skema pembentukan key yang digunakan ialah mengambil 6 karakter awal dari *password* dan 4 *digit* terakhir dari NIK *user* , sehingga akan terbentuk 10 *digit* key dengan kombinasi NIK dan *Password* . Pengujian ini perlu dilakukan untuk membuktikan apakah skema yang diinginkan benar-benar berjalan dan dapat menghasilkan cipher text dan juga *digest text* .

### 5.9.2 Prosedur Pengujian

Prosedur pengujian yang dilakukan ialah dengan menggunakan 3 buah jenis NIK dengan format karakter yang berbeda-beda . Pada pengujian pertama akan menggunakan *password* dengan format karakter huruf alphabet , pengujian kedua menggunakan format angka dan pengujian ketiga ialah dengan menggunakan symbol sebagai *password* . Untuk NIK yang digunakan pada ketiga pengujian berupa angka .

### 5.9.3 Hasil Pengujian

Tabel 5. 29 Hasil Pengujian Skema Pembentukan Key

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| NIK | *Password* | Key | Cipher Text | Digest Text |
| 1451502071110510 | arsana | arsana0510 | 00011001000011010101010100101010000101011010101110010000 | a099edde50255dd7d388f6925071c82e19fa6e0206816f972d1d97bd |
| 1451502071110511 | !@#$%^&\*() | !@#$%^0511 | 00001110010110110010000000010010101011000101011100111011 | 5a0d4e962d37cf5fac878fcb23ee249c2dc89bfc82810496f7ea61b2 |
| 1451502071110512 | 1234567890 | 1234560512 | 11101000001000100000001001000110000101011101001110110010 | d4d74182480fba4bb7274ea037899c6083c6e49fc6a53df99e2fa93d |

Tabel 5.18 merupakan hasil dari pengujian skema pembuatan *key* , nilai dari kolom key didapatkan dengan cara melakukan print terhadap variable hasil kombinasi NIK dan *password* pada android *studio* , sedangkan nilai *cipher text* dan *digest text* didapatkan dari *console* firebase yang digunakan . Dari hasil diatas dapat diambil kesimpulan bahwa pembuatan key dengan mengkombinasikan inputan berupa NIK dan *password* berjalan sesuai yang diharapkan serta skema tersebut dapat menghasilkan *cipher text* dan *digest text* sesuai yang diharapkan walaupun nilai *variable key* mengandung karakter berupa angka , huruf maupun simbol .

## 5.10 Pengujian *Avalanche Effect*

### 5.10.1 Tujuan pengujian

Tujuan pengujian *avalanche* *effect* ialah untuk membandingkan berapa banyak karakter yang berubah pada *output* ketika *input*nya dirubah . Pada jurnal yang berjudul *“Effective Implementation and Avalanche Effect of AES”,* hasil dari enkripsi atau *hash* dikatakan baik apabila perubahan 1 bit *input* menghasilkan perubahan besar pada *output* (Kumar, 2012).Pada jurnal tersebut juga dituliskan bahwa nilai *avalanche effect* yang baik ialah bernilai minimum 50%. Pengujian ini biasa dilakukan untuk mengetahui tingkat keamanan dari sebuah algoritme kriptografi kunci simetris dan juga algoritme hash.

### 5.10.2 Prosedur pengujian

Perhitungan *avalanche* *effect* memiliki rumus sebagai berikut .

(12)

AE =

Jumlah bit yang berubah pada hash

Jumlah bit keseluruhan pada hash

x 100%

Pengujian dilakukan kepada algoritme grain , SHA-3 dan kedua algoritme tersebut ketika dijalankan secara *linear*. Pada pengujian ini *input* yang digunakan ialah “arsana10” dan “arsana11” . Pada pengujian *avalanche* *effect* grain kedua nilai tersebut menjadi nilai untuk *key* , untuk SHA-3 kedua nilai tersebut menjadi *plain text*, dan ketika kedua algoritme tersebut berjalan secara *linear* kedua nilai tersebut menjadi nilai *key* untuk masukan algoritme grain . Perubahan karakter “0” dan “1” digunakan untuk melihat berapa banyak karakter yang berubah pada *output* ketika *input*nya dirubah 1 buah karakter . Setelah mendapatkan berapa bit yang berubah , barulah dihitung dengan menggunakan rumus AE (*Avalanche* *Effect*).

### 5.10.3 Hasil pengujian

Berikut ini adalah hasil pengujian *avalanche effect* dari algortima grain , SHA-3 dan juga kedua algoritmenya ketiga berjalan secara *linear* dengan menggunakan dua *input* sebagai pembanding.

1. Grain

Tabel 5. 30 Hasil pengujian *avalanche effect* untuk algoritme grain

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ***Key*** | ***Cipher text*** | **Jumlah Bit yang Berubah** | **Jumlah Bit** | ***Avalanche* *Effect*** |
| arsana10 | 01010111000010010011110000110110011010010011100111011101 | 25 | 56 | 44,6% |
| arsana11 | 11110001110001110010100001110100110101110000001110011111 |

1. SHA-3

Tabel 5. 31 Hasil pengujian *avalanche effect* untuk algorimta SHA-3

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Plaint Text** | ***Digest text*** | **Jumlah Bit yang Berubah** | **Jumlah Bit** | ***Avalanche* *Effect*** |
| arsana10 | 74797a532afc140b54d0383196789af2f6dd25edbd18410781481a4a | 220 | 224 | 98,2% |
| arsana11 | ced385d8eb1bb8724c1fa25ce9c326dfe7f5c80f55083d1d46f66499 |

1. Grain dilanjutkan dengan SHA-3

Tabel 5. 32 Hasil pengujian *avalanche effect* untuk algorimta grain dilanjutkan dengan SHA-3

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ***Key*** | ***Digest text*** | **Jumlah Bit yang Berubah** | **Jumlah Bit** | ***Avalanche* *Effect*** |
| arsana10 | d927b8753b810d72042ff278aa4628debf20c9326bbf34c7eda4be5a | 216 | 224 | 96,4% |
| arsana11 | 8262d59ba8bf2294dd4bb7a47f05eee0d93498bbdb3b4f7265ab8686 |

### 5.10.4 Analisis pengujian

Hasil pengujian *avalanche* *effect* untuk algoritme grain tercatat pada Tabel 5.19 , sedangkan algoritme SHA-3 pada tabel 5.20 dan juga ketika kedua algoritme tersebut berjalan secara *linear* pada tabel 5.21 .

Untuk pengujian *avalanche* *effect* pada algoritme grain, *cipher text* untuk kedua *key* yang berbeda berubah sebanyak 44.6 % . Pada algoritme SHA-3 memiliki nilai *avalanche* *effect* sebesar 98.2 % ketika *plain text* dirubah 1 karakter . Lalu ketika kedua algoritme tersebut dijalankan secara *linear* dengan *output* dari algoritme grain digunakan sebagai masukan algoritme SHA-3 nilai *avalanche*nya ialah 96.4 % . Dari hasil ini terlihat bahwa algoritme grain memiliki nilai *avalanche* yang sangat rendah yaitu 44.6 % , hal ini sangat wajar karena grain memiliki 3 buah *input* dan pada pengujian ini hanya di lakukan pengujian dengan 1 *input*an yang berbeda dari ketiga buah *input*an tersebut , berbeda dengan SHA-3 yang hanya memiliki 1 buah *input* dan menghasilkan nilai 98.2 % . Namun ketika kedua algoritme tersebut dijalankan secara *linear* dengan *output* grain digunakan sebagai *input* dari SHA-3 maka *avalanche* *effect*nya menjadi 96.4% . Berarti dengan melakukan hash terhadap *output* dari grain dapat memperbaiki nilai *avalanche* *effect* algortima grain, walaupun pada *input* algoritme grain hanya merubah 1 jenis *input* . Jika dibandingkan dengan nilai *avalanche* *effect* algoritme SHA-3, *avalanche* *effect* dari grain yang dilanjutkan dengan SHA-3 memiliki nilai lebih rendah , namun hal ini bukan menjadi masalah karena nilai *avalanche* *effect* minimal ialah 50 % sebagai syarat algoritme yang baik.

# BAB 6 PENUTUP

## 6.1 Kesimpulan

Berdasarkan hasil penelitian yang telah dilakukan, dapat disimpulkan beberapa hal mengenai algoritme grain dan SHA-3.

1. Algoritme grain dan algoritme SHA-3 memiliki waktu rata-rata 153,62 ms dalam memproses masukan hingga mengeluarkan keluaran yang diinginkan ketika kedua algoritme tersebut berjalan secara *linear* .
2. Algoritma grain dapat memberikan aspek confidentiality kepada user dibuktikan dengan pengujian confidentiality yang telah dilakukan , grain akan menghasilkan output berupa cipher text yang dapat dikembalikan lagi menjadi plain text , sedangkan SHA-3 dapat memberikan aspek integrity kepada user yang dibuktika dengan pengujian integrity
3. Analisis untuk ketahanan pada algoritme grain dan SHA-3 terhadap serangan dilakukan dengan cara memperhatikan dan membandingkan hasil dari *avalanche* *effect* yang dimana dari pengujian tersebut menunjukan nilai 96,4% untuk algoritme grain dan SHA-3 ketika kedua algoritme tersebut dikombinasikan .

## 6.2 Saran

Saran yang dapat penulis berikan untuk penelitian selanjutnya yaitu dapat membandingkan algoritme grain dan SHA-3 dengan kombinasi algoritme lainya dari segi waktu pemrosesan maupun untuk tingkat keamananya pada penerapan aplikasi pemilu *online* . Selain itu dapat juga dikembangkan dari sisi perangkat lunaknya agar dapat memenuhi kriteria standar aplikasi pemilu *online* baik untuk kebutuhan fungsional maupun non-fungsional .Diharapkan penelitian ini dapat digunakan sebagai rujukan untuk pengembangan selanjutnya .

# Daftar Pustaka

|  |  |
| --- | --- |
|  | Ariyus, D., 2009. *Keamanan Multimedia*. Yogyakarta : Andi  Bertoni, G .,Daemen, J.,Peeters, M.& Van Aschee,G. 2007. *Sponge function.* STMicroelectronics NXP Semiconductors.  Computer Security Resource Center National Institue of Standards and Technology Information Technolog Laboratory, 2017. *SHA-3 Standardization* [*Online*]. Tersedia:<http://csrc.nist.gov/groups/ST/hash/SHA-3/SHA-3\_standardization.html>. [Diakses 16 8 2017].  Darmawan,I.,Nurhandjati,N.,Kartini,E., 2014. *Memahami E-voting: Berkaca dari Pengalaman Negara-negara Lain dan Jembrana (Bali)*, Yayasan Pustaka Obor Indonesia.  Elminaam,D.S.A., Kader,H.M.A.& Hadhoud,M.M. 2009 *Energy Efficiency of Encryption Schemes for Wireless Devices*, *International Journal of Computer Theory and Engineering*.  Hell,M., Johansson,T. & Meier,W., 2006. *Grain - A Stream Cipher for Constrained Environments*.  Junaidi,V., 2009. *Menata Sistem Penegakan Hukum Pemilu Demokratis Tinjauan Kewenangan MK atas Penyelesaian Perselisihan Hasil Pemilu (PHPU)*. *Jurnal Konstitusi* Volume 6, Nomor 3  Kumar,A. & Tiwari,Namita., 2012. *Effective Implementation and Avalanche Effect of AES*. International Journal of Security, Privacy and Trust Management (IJSPTM), Vol. 1, No ¾  Maesaroh, *Anggaran total Pemilu 2014 Rp24,1 T*, 15 Maret 2013. [*Online*]. Tersedia: <https://nasional.sindonews.com/read/727799/12/anggaran-total-pemilu-2014-rp241-t-1363353173>. [Diakses 12 Agustus 2017].  Rainer, D. , *Pengertian Pemilu, Tujuan, Fungsi, Asas, Bentuk, Sistem Terlengkap* . 23 Oktober 2017. [*Online*]. Tersedia http://www.spengetahuan.com/2017 /09/pengertian-pemilu-tujuan-fungsi-asas-bentuk-sistem.html>.[Diakses 12 Agustus 2017].  Rhomine,C.H., 2015. *SHA-3 Standard: Permutation-Based Hash and Extendable-Output Functions*. Federal Information Processing Standards Publications  Sentot,K,. 2009 . *Teori & Aplikasi Kriptografi*. SPK IT Consulting .  Smith ,A.D. & Clark,J.S. 2005. *Revolutionizing the voting process through online strategies*., *Online Information Review,* vol. 29, no. 5, pp. 513-530.  Sommerville, I., 2003 *Software* Engineering, 9th ed. London:Addison Wesley.  Statista, *Global smartphone sales to end users from 1st quarter 2009 to 1st quarter 2017, by operating system (in million units).* 2017. [*Online*]. Tersedia: <https://www.statista.com/statistics/266219/global-*smartphone*-sales-since-1st-quarter-2009-by-operating-*system*/>. [Diakses 12 Agustus 2017].  Sukrino,D., *Menggagas Sistem Pemilihan Umum Di Indonesia*, *Jurnal Konstitusi Pusat Kajian Konstitusi Universitas Kanjuruhan Malang Nomor 1,* vol. II, p. 10, Mahkamah Kontitusi Republik Indonesia.  Votehere. Inc., 2012. *Network Voting Systems Standards*, *Public Draft 2*  Zaki,A., 2008 . *E-Life Style Memanfaatkan Beragam Perangkat Teknologi Digital*. Penerbit Salemba Infotek |
|  |  |
|  |  |
|  |
|  |  |
|  |  |