BAB 2 LANDASAN KEPUSTAKAAN

2.1 Kajian pustaka

Telah banyak penelitian yang berkaitan dengan fungsi algoritma hash. Pada Tabel 2.1 penulis mencoba menuliskan apa yang telah dipelajari untuk mendukung penelitian dalam skripsi ini. Dari jurnal yang dipelajari, hal-hal yang didapat antara lain teori dasar aplikasi berbasis *web* secara umum, kriptografi, keamanan *web*, dan juga bagaimana cara untuk menguji, membandingkan, dan menganalisis algoritma *hash* yang telah ada, sehingga dapat digunakan dalam penelitian ini.

Berdasarkan penelitian sebelumnya yang berkaitan dengan algoritma SHA-1, yaitu berjudul *Finding Collisions in the Full SHA-1* telah ditemukan kolisi pada algoritma *hash* SHA-1 (Wang, 2005). Sedangkan penelitian sebelumnya untuk algoritma SHA-3 yang berjudul *Efficient and Concurrent Reliable Realization of the Secure Cryptographic SHA-3 Algorithm*. Di sana dijelaskan tentang bagaimana algoritma Keccak menjadi pemenang dan distandarisasi menjadi algoritma SHA-3, serta kelebihannya dibandingkan algoritma lain (Sarmadi, 2014).

Pada pustaka yang telah dipelajari, kebanyakan penelitian sebelumnya terfokus pada algoritma *hash* yang sudah lama, seperti MD5 dan SHA-1. Dari hal itulah yang melatarbelakangi penelitian tentang algoritma SHA-3 yang terbilang masih baru, bagaimana kinerjanya pada saat implementasi dan bagaimana jika dibandingkan dengan pendahulunya yaitu SHA-1.

Dengan berbekal metode dari penelitian sebelumnya, diharapkan dapat membangun sebuah sistem yang lebih aman dengan memanfaatkan metode atau solusi yang sudah diteliti tersebut yaitu SHA-3 agar terbangun sebuah sistem yang dapat diandalkan baik dari segi integritas data.

Tabel 2.1 Perbandingan penelitian penulis dengan penelitian terkait

No	Judul	Objek	Metode	Hasil
1	*Finding Collisions in the Full SHA-1* (Xiayou Wang, 2005)	Algoritma *hash* SHA-1	- Analisis kompleksitas - Teknik modifikasi pesan - Metode pencarian kolisi dengan menggunakan kolisi yang terdekat - Teknik untuk membangun	- Telah berhasil dilakukan serangan pertama pada algoritma SHA-1 dengan kompleksitas kurang dari 2^{69} operasi *hash* - Pemrosesan pesan yang lebih rumit dapat memberikan keamanan yang lebih baik - Diharapkan analisis algoritma SHA-1 dapat berguna untuk penelitian selanjutnya

			jalur diferensial	
2	*Efficient and Concurrent Reliable Realization of the Secure Cryptographic SHA-3 Algorithm* (Siavash Bayat-Sarmadi, 2014)	Algoritma *hash* SHA-3	- Pendekatan berbasis RERO - Skema pendeteksi kesalahan	- Menghasilkan sebuah skema *time-redundancy* untuk pendeteksi kesalahan untuk algoritma kriptografi SHA-3 - Skema pendeteksi kesalahan tersebut memberikan manfaat pada saat implementasi hardware dari algoritma SHA-3 lebih dapat diandalkan
3	Analisis dan Implementasi Algoritma SHA-1 dan SHA-3 pada Sistem Keamanan Autentikasi Garuda Training Cost (Firlhi Kurniawan, 2017)	Algoritma *hash* SHA-1 dan SHA-3	- Analisis terhadap fase login aplikasi untuk mengetahui processing time - Pengujian dengan metode *brute-force* untuk kedua algoritma yaitu SHA-1 dan SHA-3	- Mengetahui algoritma manakah yang lebih baik, SHA-1 ataukah SHA-3 - Melakukan penggantian pada sistem keamanan aplikasi yang sebelumnya menggunakan algoritma *hash* SHA-1 menjadi SHA-3 - Mengetahui kekurangan dan kelebihan dari masing masing algoritma

2.2 *Garuda Training Cost*

Garuda Training Cost merupakan suatu aplikasi berbasis *web* yang dikembangkan di Garuda Indonesia Training Center. Aplikasi ini digunakan untuk membantu perhitungan biaya pelatihan pilot, pramugari, dan staff di Garuda Indonesia Training Center. Aplikasi ini dibuat karena komponen pada biaya tersebut sangat banyak, dan untuk mempermudah maka dibuatkanlah sistem ini. Terdapat juga fitur untuk perhitungan keuntungan yang akan didapatkan dan fitur untuk membuat suatu laporan keuangan secara otomatis ketika ada suatu instansi yang melakukan pendaftaran pelatihan. Aplikasi ini menggunakan prinsip *client-server*, dimana ada terdapat 3 jenis pengguna, yaitu *admin*, *marketing*, dan *training analyst.*

Untuk menjamin keamanan data yang terdapat pada *database*, maka ada beberapa sistem keamanan yang telah diterapkan pada sistem *Garuda Training Cost* ini, antara lain:

a. *Session*

   Session merupakan salah satu fasilitas yang terdapat pada bahasa pemrograman PHP yang digunakan untuk menyimpan data sementara ke dalam variabel, sehingga data tersebut dapat diakses oleh *client*. *Session* biasanya digunakan untuk menyimpan id pengguna. Nilai variabel pada *session* disimpan di sisi *server*. *Session* juga digunakan untuk mencegah seseorang yang belum *login* untuk mengakses kedalam sistem.

b. *Password hash* dengan SHA-1

   Semua *password* dari pengguna yang telah terdaftar akan tersimpan di dalam *database*. Tanpa sistem keamanan, maka *password* yang tersimpan didalam *database* dapat dilihat atau hanya berbentuk *plaintext.* Jika seorang *cracker* dapat menembus sistem keamanan aplikasi tersebut, maka dengan mudahnya *cracker* tersebut dapat mengetahui password dari pengguna yang ada di dalam database. Maka dari itu, digunakanlah algoritma hash SHA-1 untuk setiap *password* yang disimpan di *database*, untuk mencegah seseorang yang tidak berwenang mengetahui *password* tersebut.

c. Pengguna *multi-level* dengan *single login*

   Dikarenakan pada aplikasi *web Garuda Training Cost* ini terdapat 3 jenis pengguna, yaitu *admin*, *marketing* dan *training analyst*, maka dibuatkanlah fasilitas ini. Hal yang dapat dilakukan ketiga jenis pengguna tersebut berbeda-beda, jadi ketika pengguna *login*, dia akan diarahkan langsung berdasarkan level hak aksesnya. Jadi pengguna tersebut tidak bisa mengakses halaman untuk pengguna jenis lain. Misalnya A adalah *admin*, dia login sebagai *admin* maka akan diarahkan ke halaman *admin*. Jika dia mencoba untuk mengakses halaman untuk *marketing* atau *training analyst*, maka akan diarahkan kehalaman yang berisi pesan anda tidak mempunyai akses untuk mengakses halaman ini.

2.3 Keamanan

Keamanan pada komputer adalah perlindungan yang diberikan ke sistem informasi otomatis untuk mencapai tujuan yang berlaku untuk menjaga integritas, ketersediaan, dan kerahasiaan sumber daya sistem informasi (termasuk perangkat keras, perangkat lunak, *firmware*, informasi/data dan telekomunikasi) (Stalling, 2011). *World Wide Web* (WWW) merupakan sebuah aplikasi *client/server* yang berjalan di *layer* TCP/IP. Jadi keamanan untuk *web* masih termasuk dalam lingkup keamanan pada komputer. Tetapi keamanan yang dibutuhkan oleh *web* berbeda dengan keamanan komputer pada umumnya, maka ini bisa menjadi topik yang menarik dalam pengembangan sistem keamanan *web*.

Terdapat enam standar keamanan secara umum yang harus dipenuhi juga oleh suatu sistem keamanan web, antara lain:

a. *Confidentially*: data-data yang tersimpan pada aplikasi *web* harus terjaga kerahasiaannya
b. *Authentication*: yang bisa mengakses aplikasi *web* dan *web server* hanyalah orang yang berhak saja
c. *Integrity*: semua data dan informasi yang tersimpan pada aplikasi *web* harus terjamin kebenaran datanya, tidak boleh dimodifikasi dari pihak yang tidak berhak
d. *Nonrepudiation*: pengguna tidak dapat menyangkal infomasi yang dimilikinya pada aplikasi *web*, dikarenakan aplikasi tersebut dapat menunjukan identitas pemilik dari data tersebut kepada pengguna
e. *Access control*: pembatasan akses terhadap aplikasi *web* dan *web server* nya, sehingga hanya pihak yang hanya memiliki akses lah yang dapat menggunakannya
f. *Availability*: memastikan aplikasi tersebut dapat diakses oleh pengguna ketika pengguna mengaksesnya

Untuk menerapkan keamanan maksimal pada sistem keamanan *web*, enam standar keamanan pada jaringan tersebut haruslah terpenuhi. Dalam penelitian kali ini, aspek keamanan yang akan ditekankan yaitu *integrity*, dikarenakan akan dilakukan penelitian terhadap algoritma SHA-1 dan SHA-3 untuk integritas dan keamanan dari *password* yang terdapat dalam sistem.

2.4 *Hash function*

Fungsi *hash* atau *hash function* adalah suatu fungsi yang memetakan suatu pesan dengan panjang yang berbeda dan memproduksi keluaran dengan panjang tetap, yang dinamakan *hash code* (Stalling, 2005). *Hash code* merupakan fungsi dari seluruh bit dari pesan dan menyediakan kemampuan untuk mendeteksi kesalahan. Perubahan dari setiap bit dari pesan akan menghasilkan perubahan pada *hash code.*

Perbedaan hash dengan algoritma enkripsi simetrik/asimetrik yaitu, hash tidak memerlukan kunci untuk melakukan enkripsi. Dikarenakan pada hash function tidak akan dilakukan dekripsi. *Hash code* tadi merupakan hasil dari pemrosesan *function*, dimana jika perubahan terjadi pada isi pesan tersebut, maka *hash code* yang dihasilkan akan berbeda.

Dalam buku yang berjudul *"Cryptography and Network Principles and Practices",* menyatakan terdapat dua kategori serangan untuk keamanan fungsi *hash,* yaitu *brute-force attacks* dan *cryptanalysis* (Stalling, 2005). Ketahanan dari suatu *hash function* bergantung kepada seberapa panjang dan rumit *hash code* yang dihasilkannya. Sedangkan cara kedua yaitu kriptanalis, mereka memanfaatkan sisi kelemahan logika pada algoritma *hash* yang digunakan

Untuk meningkatkan keamanan pada fungsi *hash*, biasanya penggunaan suatu algoritma digabung dengan *salt. Salt* adalah suatu pengacak tambahan dengan nilai tertentu untuk digabungkan dengan *hash code*. Nilai dan fungsi dari *salt* dapat dengan bebas ditentukan. Dengan begitu, *digest* akan lebih tahan terhadap serangan *brute-force*.

Terdapat berbagai macam algoritma *hash function* yang dikembangkan saat ini. Tetapi algoritma yang paling umum digunakan yaitu SHA-1. SHA-3 masih terbilang baru dan akan dilakukan pengujian kinerjanya di penelitian ini.

2.4.1 *Secure Hash Algorithm-1* (SHA-1)

SHA dikembangkan oleh *National Institute of Standards and Technology (NIST)* dan dipublikasikan sebagai *Federal Information Processing Standards (FIPS 180)* pada tahun 1993. *Secure Hash Standard (SHS)* menspesifikasikan SHA-1 untuk menghitung nilai *hash* dari sebuah pesan atau *file*. SHA-1 memiliki panjang pesan maksimal 2^{64} bits dan memiliki keluaran sebesar 160 bit yang dinamakan *message digest* atau *hash code. Message digest* tersebut dapat digunakan sebagai masukan untuk *Digital Signature Algorithm (DSA),* yang digunakan untuk menghasilkan *signature* untuk memverifikasi pesan tersebut.

Pada saat ditemukan kelemahan pada algoritma SHA-0, berbagai revisi dan perbaikan dilakukan untuk membuat suatu algoritma yang lebih baik. Hasil dari perbaikan tersebut diterbitkan pada tahun 1995 dan dijadikan acuan untuk pembuatan algoritma SHA-1.

Algoritma SHA-1 merupakan revisi teknis dari algoritma SHA. Algoritma SHA-1 dapat dikatakan aman karena proses perhitungannya tidak memungkinkan untuk menemukan pesan yang sebenarnya dari *message digest* yang dihasilkan. Setiap perubahan yang terjadi pada pesan saat perjalanan akan menghasilkan *message digest* yang berbeda. Algoritma SHA-1 berbasis pada algoritma MD4 dan rancangannya sangatlah mirip terhadap algoritma tersebut.

Berikut ini merupakan cara kerja dari algoritma SHA-1 dalam menghasilkan *message digest* merujuk buku yang berjudul *"Internet Security Cryptographic principles, algorithms and protocols'* (Rhee, 2003).

a. Message padding

Message padding merupakan proses penambahan angka 0 sampai panjangnya mencapai 512 bits. SHA-1 akan memproses blok dengan panjang 512 bits tersebut secara sekuensial pada saat menghitung nilai *hash* dari sebuah pesan atau file yang diberikan sebagai masukan. *Padding* pada algoritma ini sama seperti yang dilakukan pada algoritma MD5. Langkah-langkah pada proses *padding* ini adalah sebagai berikut. Misalkan pesan yang akan diproses adalah 'abc', huruf tersebut dikonversi terlebih dahulu ke dalam bentuk biner, setelah itu ditambahkan angka '1' pada biner tadi. Setelah itu dikonversi kembali kedalam bentuk heksadesimal, kemudian ditambahkan angka '0' sampai panjangnya mencapai 512 bits. Kemudian dihitunglah panjang dari biner tadi setelah ditambahkan angka '1', yang hasilnya adalah 24 bit dan 25 bit setelah ditambahkan angka '1'. Setelah itu kita menambahkan '0' sebanyak 423 dan nilai heksadesimal dari 24, yaitu 18. Pesan yang telah dilakukan *padding* berisi satu blok yang berisi 16 *words.* 16 x 8 x 4 = 512 bits untuk n = 1 pada kasus ini.

b. Inisialisasi *buffer* sebesar 160 bit

Buffer sebesar 160 bit ini terdiri dari 5 buah *register* 32 bit (A, B, C, D dan E). Sebelum memproses blok, *register* tersebut di inisialisasi terlebih dahulu dengan menggunakan nilai heksadesimal sebagai berikut.

H_0 = 67 45 23 01

H_1 = ef cd ab 89

H_2 = 98 ba dc fe

H_3 = 10 32 54 76

H_4 = c3 d2 e1 f0

c. Fungsi yang digunakan

Terdapat beberapa fungsi logika yang digunakan pada algoritma SHA-1. Setiap fungsi dari perulangan ke 0 sampai perulangan ke 79 mengoperasikan tiga buah *word* sebesar 32 bit*,* yaitu B, C dan D, dan memproduksi sebuah sebuah *word* sebesar 32 bit sebagai keluaran. Setiap operasi melakukan operasi non-linier dari A, B, C, dan D, kemudian dimasukkan dan ditambahkan. Fungsi-fungsi SHA tersebut didefinisikan sebagai berikut.

f_t (B, C, D) = (B • C) + ($\overline{B}$ • D), 0 <= *t* <= 19

f_t (B, C, D) = B $\oplus$ C $\oplus$ D, 20 <= *t* <= 39

f_t (B, C, D) = (B • C) + (B • D) + (C · D), 40 <= *t* <= 59

f_t (B, C, D) = B $\oplus$ C $\oplus$ D, 60 <= *t* <= 79

Keterangan:

B • C = melakukan operasi AND dari B dan C

B $\oplus$ C = melakukan operasi XOR dari B dan C

B = melakukan operasi komplemen dari B

+ = penambahan modulo 2^{32}

d. Konstanta yang digunakan

Terdapat empat konstanta yang digunakan pada algoritma SHA-1. Konstanta tersebut didefinisikan sebagai berikut.

K_t = 5a827999, digunakan pada perulangan 0 <= t <= 19

K_t = 6ed9eba1, digunakan pada perulangan 20 <= t <= 39

K_t = 8fbbcdc, digunakan pada perulangan 40 <= t <= 59

K_t = ca62c1d6, digunakan pada perulangan 60 <= t <= 79

e. Perhitungan *message digest*

Message digest dihitung menggunakan pesan yang telah dilakukan *padding* sebelumnya. Untuk menghasilkan *message,* blok yang berisi 16 *word* (M_0 sampai M_{15}) diproses secara berurutan. Setiap M_i dilakukan perulangan sebanyak 80 kali dengan menggunakan 32 bit word (W_0 sampai W_{79}) dengan menggunakan algoritma sebagai berikut. Pertama dilakukan pemisahan dulu terhadap M_i menjadi 16 *word,* W_0 sampai $W_{79.}$

Pseudocode

For t = 0 to 15, $W_t = M_t$

For t = 16 to 79, $W_t = S^1(W_{t-16} \oplus W_{t-14} \oplus W_{t-8} \oplus W_{t-3})$

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

For t = 0 to 79 do

TEMP = S^5 (A) + F_t(B, C, D) + E + W_t + K_t;

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

Keterangan:

A, B, C, D, E = lima *word* dari *buffer*

t = angka perulangan, 0 <= *t* <= 79

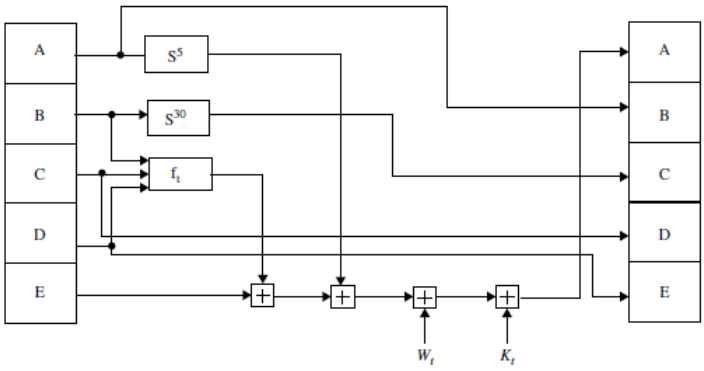
S^i = pergeseran ke kiri secara *circular* sebanyak I bits

W_t = *word* berukuran 32 bit yang didapatkan dari blok masukan 512 bit

K_t = konstanta

+ = penambahan modulo 2^{32}

Setelah keseluruhan blok berukuran 512 bit selesai diproses, keluaran dari perulangan terakhir merupakan *message digest,* yang direpresentasikan dari lima *words,* yaitu H_0, H_1, H_2, H_3, H_4. Gambar 2.1 adalah operasi dari algoritma SHA-1 yang digambarkan dalam bagan.



Gambar 2.1 Operasi SHA-1

Semua dokumen atau pesan dalam panjang berapapun dapat menghasilkan *message digest* berukuran 160 bit yang diproduksi menggunakan algoritma SHA-1. Dalam buku ini juga disebutkan tidak mungkin ditemukan pesan yang memiliki *digest* yang sama.

Tetapi, algoritma SHA-1 ini sudah kurang aman karena sudah ditemukannya cara manual untuk menemukan kolisi pada algoritma ini (Wang, 2005). Lebih dipertegas lagi oleh jurnal yang berjudul *"Finding SHA-1 Characteristics: General Results and Applications".* Dalam tulisan tersebut mereka sudah dapat membuat sebuah metode otomatis untuk menemukan kolisi tersebut dan berhasil dilakukan pada beberapa pesan yang berbeda (Rechberger, 2006).

2.4.2 *Secure Hash Algorithm-3* (SHA-3)

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 algoritma *hash* alternatif yang sangat berbeda dengan algoritma sebelumnya, yaitu SHA-3.

Pada 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 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. Dikarenakan berbagai jenis keluaran tersebut, maka SHA-3 ini masuk kedalam jenis *sponge function.* Menurut pembuat dari algoritma SHA-3 ini, pada jurnalnya yang berjudul *"Cryptographic sponge function", sponge function* menyediakan cara tertentu agar dapat menggeneralisasikan fungsi *hash* dengan hasil keluaran yang beragam (Bertoni, 2011).

Berdasarkan penjelasan dari publikasi resmi 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.

a. Inisialisasi *input* dan *output*
   Pada fase ini, akan dilakukan pemilihan jenis algoritma SHA-3 mana yang akan digunakan. Jenis-jenis algoritma 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 "password". Masukan tersebut akan diproses, setelah diproses maka akan dihasilkanlah *digest* dari *message* tadi.

b. *Absorbing*
   Pada fase ini, *block state* dari *plaintext* yang telah diproses pada fase sebelumnya tadi akan dimasukkan kedalam algoritma dan kemudian diproses lagi. Proses ini juga melibatkan fungsi permutasi Keccak.
c. *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.
d. Permutasi Keccak
   Permutasi Keccak bisa dibilang "jantung" dari algoritma ini. Semua fase yang dilalui harus melalui fungsi ini dulu. Permutasi Keccak dibagi menjadi 5 tahapan, yaitu θ *(Theta)*, ρ *(Rho)*, π *(Phi)*, χ *(Chi)* 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.
   Untuk setiap tahapan, akan dilakukan dalam perulangan yang dilakukan 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

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

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

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

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]

2.4.3 Perbandingan SHA-1 dengan SHA-3

1. Algoritma SHA-1 hanya dapat mengolah input dengan panjang maksimal 2^{64} bit, sedangkan algoritma SHA-3 dapat mengolah input dengan panjang tak terhingga.
2. Ukuran *output* yang dihasilkan oleh algoritma SHA-1 berukuran 160 bits. Sedangkan ukuran output yang dihasilkan SHA-3 berukuran beragam, mulai dari 224, 256, 384, dan 512 bits. Menjadikan algoritma SHA-3 lebih tahan terhadap serangan *brute-force*.
3. Pada tahun 2015 telah ada yang berhasil melakukan serangan *theoretical* dan *collision* pada algoritma SHA-1, menjadikan algoritma ini sudah kurang aman jika dibandingkan SHA-3 yang masih sangat baru dan belum bisa ditembus.

2.5 *Brute-force attack*

Brute-force attack merupakan salah satu jenis metode serangan. Dari hasil studi literatur yang dilakukan, pada jurnal yang berjudul *"Understanding brute-force",* cara kerja *brute-force* yaitu dengan melakukan *trial-and-error* yang digunakan untuk mendapatkan suatu informasi yang belum diketahui (Bernstein, 2005), seperti *username* dan *password*. Proses *brute-force* ini menggunakan bantuan sebuah aplikasi yang bekerja secara otomatis, yang mencoba semua kemungkinan yang ada untuk dapat menemukan *username* dan *password* tadi. Semakin baik algoritma yang digunakan, maka akan semakin lama waktu yang dibutuhkan untuk melakukan serangan *brute-force* pada hash tersebut (Shaugi, 2012). Proses ini memakan waktu yang sangat lama, di karenakan kombinasi *username* dan *password* yang sangat banyak. Tetapi seiring perkembangan *software* dan *hardware* yang sangat cepat, proses *brute-force* dapat dilakukan dalam waktu yang lebih singkat.

2.6 *Independent Samples t-Test*

Independent samples t-test merupakan pengujian yang digunakan untuk membandingkan perbedaan antara *means* atau rata-rata dari dua kelompok independen untuk menentukan apakah ada bukti statistik yang menunjukkan rata-rata dari populasi tersebut memiliki perbedaan yang signifikan. Pada umumnya *independent samples t-test* disebut dengan rancangan antar kelompok, dan juga dapat digunakan untuk menganalisis kelompok control dan eksperimental.

Pada *independent sample t-test*, setiap kasus harus memiliki dua kategori, yang pertama adalah *grouping variable* dan *dependent variable. Grouping*

variable membagi kasus menjadi dua kelompok yang saling berhubungan, seperti laki-laki atau perempuan untuk *grouping variable* untuk jenis kelamin, sedangkan *test/dependent variable* menjelaskan nilai dari dimensi kuantitatif misalnya performa. *T-test* mengevaluasi apakah nilai rata-rata dari *test variable* untuk kelompok jenis kelamin laki-laki memiliki perbedaan yang signifikan dengan nilai rata-rata dari *test variable* untuk kelompok jenis kelamin perempuan (Oak, 2006).

Berikut ini akan disajikan contoh tabel *independent sample t-test* dan cara untuk membacanya.

		Levene's Test for Equality of Variances	
		F	Sig.
Waktu	Equal variances assumed	128,646	,000
	Equal variances not assumed		

Tabel 2.2 Tabel *Levene's Test for Equality of Variances* untuk waktu

Tabel 2.2 merupakan tabel *Levene's Test for Equality of Variances* yang digunakan untuk menguji asumsi terhadap homogenitas dari suatu varians. Untuk Tabel 2.2 yang menunjukkan Waktu, nilai F nya adalah 128,646 dan nilai Sig. nya adalah 0,000. Karena nilai Sig. pada pengujian tersebut kurang dari nilai *alpha* yaitu 0,05 (p < 0,05), maka hipotesis yang pertama atau H0 yang ditunjukan dengan baris *"Equal variances assumed"* ditolak, yang berarti dari asumsi tersebut dapat disimpulkan bahwa terdapat perbedaan yang signifikan antara varians dari kedua kelompok. Maka hipotesis yang digunakan untuk pengujian pada tabel *t-test for Equality of Means* pada Tabel 2.3 yaitu H1 yang ditunjukan dengan baris *"Equal variances not assumed".*

t-test for Equality of Means						
					95% Confidence Interval of the Difference	
t	df	Sig. (2-tailed)	Mean Difference	Std. Error Difference	Lower	Upper
-47,180	58	,000	-652,500	13,830	-680,184	-624,816
-47,180	29,160	,000	-652,500	13,830	-680,779	-624,221

Tabel 2.3 Tabel *t-test for Equality of Means* untuk waktu

Tabel 2.3 merupakan tabel *t-test for Equality of Means* yang digunakan untuk menguji apakah ada perbedaan yang signifikan antara *means* atau rata-rata dari kedua kelompok. Pada Tabel 2.3 nilai t nya adalah -47,180 dengan nilai Sig. 0,000. Karena nilai Sig. pada pengujian tersebut kurang dari nilai *alpha* yaitu 0,05 (p < 0,05), maka hipotesis yang pertama atau H0 ditolak, yang berarti dapat disimpulkan bahwa terdapat perbedaan yang signifikan pada kedua kelompok tersebut.

2.7 *One-Way ANOVA*

One-way Analysis of Variance (ANOVA) merupakan sebuah prosedur untuk menguji apakah hipotesis dari *means* atau rata rata dari *K* populasi adalah sama, dimana nilai *K* >= 2. *One-way ANOVA* membandingkan rata-rata dari beberapa sampel atau kelompok yang bertujuan untuk mendapatkan kesimpulan dari rata-rata populasi tersebut. *One-way ANOVA* juga biasa disebut dengan *single factor analysis of variance* karena hanya terdapat satu variabel independen atau faktor.

Dalam *ANOVA*, terdapat dua jenis variabel, yaitu variabel independen dan variabel dependen. Variabel independen dikendalikan atau dimanipulasi oleh peneliti. Variabel ini digunakan untuk membentuk sebuah pengelompokkan dari penelitian. Pada *One-way ANOVA,* hanya ada satu variabel independen yang dipertimbangkan, tetapi terdapat dua atau lebih tingkatan dari variabel independen. Biasanya variabel independen merupakan variabel yang dapat dikategorikan. Variabel independen membagi individual ke dalam dua kelompok atau lebih. Terdapat dua jenis variabel independen, yaitu aktif dan atribut.

Variabel dependen didefinisikan sebagai variabel sebagai hasil manipulasi variabel independen. Pada *One-way ANOVA,* hanya terdapat satu variabel dependen, dan hipotesis didapatkan dari *means* atau rata-rata dari variabel dependen tersebut. Variabel dependen membedakan individu pada beberapa dimensi kuantitatif (Oak, 2006).

Berikut ini akan disajikan contoh tabel *One-way ANOVA* dan cara untuk membacanya.

ANOVA

Waktu

	Sum of Squares	df	Mean Square	F	Sig.
Between Groups	5,747E+10	2	2,874E+10	640,500	,000
Within Groups	3903301689	87	44865536,65		
Total	6,138E+10	89			

Tabel 2.4 Tabel *One-way ANOVA* untuk waktu

Langkah pertama yang dilakukan pada pengujian *One-way ANOVA* adalah menguji asumsi homogenitas pada varians, dimana hipotesis pertama atau H0 mengasumsikan bahwa tidak terdapat perbedaan antara varians dari kelompok. Untuk variabel waktu yang ditunjukan pada Tabel 2.4, nilai F dari *Levene's test* adalah 640,500 dengan nilai Sig. 0,000. Karena nilai Sig. kurang dari nilai *alpha* yaitu 0,05 ($p < 0,05$), maka hipotesis pertama atau H0 ditolak, yang berarti terdapat perbedaan yang signifikan antara varians dari kedua atau lebih kelompok tersebut.