IMPLEMENTASI ALGORITMA RC6 DAN METODE ONE TIME PASSWORD UNTUK MEMBANGKITKAN KODE OTENTIKASI WEBSITE E-VOTING MELALUI SMS

SKRIPSI

Diajukan Untuk Memenuhi Bagian Dari
Syarat Memperoleh Gelar Sarjana Komputer
Pada Departemen Pendidikan Ilmu Komputer
Program Studi Ilmu Komputer

oleh

Yogi Siswanto

1301614

PROGRAM STUDI ILMU KOMPUTER

DEPARTEMEN PENDIDIKAN ILMU KOMPUTER

FAKULTAS PENDIDIKAN MATEMATIKA DAN ILMU PENGETAHUAN ALAM

UNIVERSITAS PENDIDIKAN INDONESIA

2018

Daftar Tabel

Daftar Isi

Daftar Gambar

BAB I
PENDAHULUAN

Dalam bab ini akan dibahas latar belakang dilaksanakannya penelitian, rumusan masalah, batasan masalah, tujuan penelitian, dan sistematika penulisan.

1.1 Latar Belakang

Dengan ditemukannya teknologi internet, para pengguna komputer semakin dimudahkan untuk berkomunikasi antara satu komputer dengan komputer lainnya. Dengan internet, komunikasi yang tadinya jauh bisa menjadi lebih dekat, yang tadinya membutuhkan waktu yang lama jadi semakin lebih cepat, yang tadinya membutuhkan biaya yang mahal menjadi semakin murah. Komunikasi dengan internet bisa dilakukan oleh siapa saja, kapan saja dan di mana saja tanpa dibatasi oleh ruang, waktu, dan biaya. Seiring dengan berevolusinya teknologi internet, maka perubahan komunikasi semakin beragam, yang tadinya komunikasi hanya sekedar melalui suara dan pesan saja, sekarang bertambah dengan adanya teknologi *video call*, pesan suara, gambar dan teknologi lainnya.

Selain itu, penggunaan internet pun menjadi semakin berkembang dikarenakana adanya pembuatan berbagai macam-macam aplikasi seperti media social dan *e-commerce*. Menurut survey yang dilakukan oleh *We are Social* pada tahun 2017, ada 3811 miliar atau sekitar 51% pengguna internet diseluruh dunia dengan 2907 miliar atau sekitar 39% menggunakan media sosial. Adapun media sosial yang populer sampai dengan awal tahun 2017 adalah *Facebook* dengan total pengguna 1968 juta pengguna, *Whatsapp* dengan total pengguna 1200 juta pengguna, *Facebook Messenger* dengan total pengguna 1000 juta pengguna, *Instagram* dengan total pengguna 600 juta pengguna.

Pada tahun 2016, *statista.com* mencatat bahwa 1,61 miliar transaksi *e-commerce* diseluruh dunia. Pada tahun yang sama pula, pendapatan dari produk retail pada *e-commerce* mencapai 1.9 milyar dollar Amerika dan juga pada tahun 2020 pendapatan dari penjualan produk retail diprediksi mencapai 4.06 milyar

dollar Amerika. Selama bulan maret tahun 2016, 46% dari dari pengguna internet yang berasal dari wilayah Asia Pasifik dan 28% pengguna internet yang berasal dari Amerika Utara melakukan transaksi pembelian secara *online*. Pada quartal ke-4 tahun 2016, penggunaan PC masih mendominasi transaksi *online* akan tetapi transaksi yang menggunakan *smartphone* berada pada posisi pertama untuk kunjugan website ritel. Survey yang dilakukan selama tahun 2017 mencatat bahwa ada sekitar 11% dari penjualan yang dilakukan menggunakan *smartphone* setiap minggunya.

Dari data-data yang didapatkan dapat menunjukkan bahwa penggunaan internet saat ini sangat pesat pertumbuhnannya dan akan bertambah seiring berjalannya waktu. Penggunaan internet saat ini didominasi oleh aplikasi-aplikasi yang berabasiskan website dan Aplikasi *Mobile*. Untuk menggunakan aplikasi sepert sosial media dan *e-commerce* diperlukannya sebuah akun yang sesuai data pribadi dan telah didaftarkan sebelumnya. Tujuan dari pendaftaran akun tersebut untuk memudahkan pengiriman barang yang telah dibeli secara *online* dan memudahkan kita untuk terhubung dengan teman kita.

Untuk melindungi data pribadi kita dari penggunaan secara ilegal, maka diperlukan sebuah metode untuk melindungi data-data tersebut. Otentikasi adalah sebuah pengaturan yang sesuai untuk mengidentifikasi dan memverifikasi bahwa pengguna yang bersangkutan telah masuk kedalam sebuah aplikasi sesuai dengan legalitas data yang berlaku.

Metode yang paling umum digunakan untuk otentikasi adalah dengan memasukkan nama pengguna atau *email* serta dibarengi dengan kode password. Hal tersebut sangat rentan terhadap serangan seperti *Shoulder Surfing*, *guesing attack*, *dictionary attack*, serta serangan *Trojan*. Beberapa penelitian mengungkapkan bahwa banyak pengguna biasa yang menggunakan kode password yang pendek serta mempunyai keterkaitan dengan nama ataupun tanggal lahir. Sayangnya, kode password seperti ini sangat mudah dipecahkan

dengan metode *Social Engineering*. Kode password yang panjang dapat membuat sistem menjadi lebih aman, akan tetapi akan sulit untuk diingat.

Selain dari serangan diatas, ada banyak serangan terhadap otentikasi seperti program mencurigakan yang merekam segala aktivitas yang dilakukan pada *keyboard* pada saat memasukan password untuk otentikasi kemudian informasi tersebut dapat dikirimkan kepada si penyerang (*Key Logging* dan *Pishing*). Untuk menghindari serangan tersebut, dibuatlah sebuah konsep *keyborad virtual* yang dapat mencegah memasukkan inputan melalui *keyboard* digantikan dengan penggunaan *mouse*.

Keamanan dari data-data pribadi menjadi hal yang sangat diperlukan. terlebih lagi pada era globalilasi sekarang ini, dimana data-data pribadi sudah mulai direkam secara elektronik. Data-data pribadi tersebut sangat rentan untuk disalah gunakan oleh pihak-pihak yang tidak bertanggung jawab. Salah satu contoh dari data pribadi yang telah direkam secara elektronik yaitu Data Pemilih Tetap yang dikelola oleh Komisi Pemilihan Umum Republik Indonesia. Data Pemilih Tetap merupakan data-data dari warga negara Indonesia yang sudah tervalidasi oleh Komisi Pemilihan Umum untuk melakukan pemilihan kepala pemerintahan ataupun wakil rakyat. Untuk mencegah kebocoran data-data tersebut, hendaknya pihak yang membuat sistem tersebut perlu untuk membangun sebuah mekanisme keamanan yang dapat melindungi Data Pemilih Tetap tersebut. Salah satu mekanisme / metode untuk melindungi Data Pemilih Tetap, adalah dengan menggunakan metode *One Time Password*. Metode *One Time Password* ini dapat diterapkan pada saat Pemilih melakukan *Login* untuk memvalidasi ulang data-data pribadi miliknya. Selain itu, metode OTP ini bisa juga diterapkan pada saat Adminstator dari pihak Komisi Pemilihan Umum melakukan *login* pada aplikasi yang berhubungan dengan Data Pemilih Tetap. Penggunaan metode ini bertujuan untuk menghindari serangan yang dilakukan oleh pihak yang tidak bertanggung jawab pada saat *login* aplikasi.

Dengan bertambahnya aplikasi yang menggunakan teknik otentikasi dan berkembangnya serangan terhadap serangan otentikasi menyebabkan berkembangnya teknologi OTP. Ide dari penggunaan OTP sebagai teknik otentikasi pertama kali dikenalkan oleh Leslie Lamport (Lamport, 1981). *One Time Password* dikenal juga sebagai enkripsi *Verman* atau algoritma enkripsi sempurna yang mana *Plaintext* digabungkan dengan kunci acak. Teknik itu merupakan salah satu metode yang diketahui berasal dari enkripsi matematika yang tidak bisa diselesaikan..

OTP merupakan algoritma simetris yang menggunakan kunci yang sama untuk proses enkripsi dan dekripsi. Proses enkripsi dan dekripsi pada teknik ini biasanya menggunakan operasi XOR untuk men-*generate* berkas enkripsi. Bruce Scheiner dalam (Schneier, 2007) mengatakan bahwa OTP merupakan skema enkripsi yang sempurna dengan kondisi jika kuncinya benar-benar acak dan tidak pernah digunakan lagi.

Selain itu One Time Password adalah sistem otentikasi dimana password hanya bisa digunakan satu kali dan pengguna harus menggunakan password baru setiap kali melakukan otentikasi. Metode ini memberikan garansi keamanan meskipun penyerang mendapatkan password pada jaringan internet atau dicuri dari pengguna. Selain itu, OTP memiliki beberapa fitur seperti kerahasiaan, mudah dibawa, dan memiliki cakupan yang sangat luas (Cho, Lee, Lee, & Lim, 2009).

Pengiriman kode otentikasi OTP, biasanya menggunakan layanan SMS dan *email*. Pengiriman kode otentikasi tersebut dilakukan dengan beberapa metode seperti *Self-updating OTP-based authentication* dan *time synchronized* OTP (Vinh, Bouzefrane, Farinone, Attar, & Kennedy, 2015).

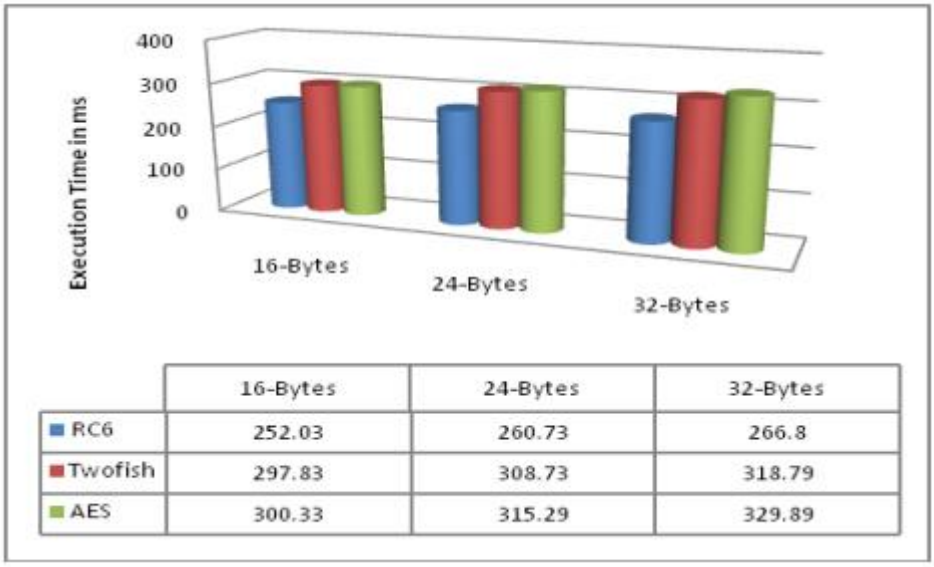
Salah satu cara yang umum untuk mengirimkan kode OTP kepada pengguna adalah melalui SMS OTP. Pengguna diharuskan untuk mengisi *username* dan *password* pada saat proses *login.* Kemudian, *username* dan *password* tersebut akan dikirmkan ke *server* untuk dienkripsi dan menghasilkan

kode OTP. *Server* menghasilkan kode OTP menggunakan algoritma kriptografi tertentu. Setelah proses enkripsi selasai, kode OTP dikirimkan kembali kepada pengguna melalui layanan SMS. Pengguna diharuskan mengisi kode OTP yang tertera pada SMS ke aplikasi untuk nantinya dikirimkan kembali ke *server*. Otentikasi terjadi pada saat server mengenali bahwa pengguna memasukkan kode yang benar saat login. Setelah kode OTP diterima oleh *server*, kode tersebut didekripsi dan kemudian dicocokan dengan kode OTP yang belum dienkripsi sama sekali. Dengan menggunakan layanan SMS ini dapat mengurangi biaya dari penggunaan perangkat keras tambahan. Selain itu, pada penelitian yang dilakukan (Wu, Garfinkel, & Miller, 2004) menunjukkan bahwa layanan SMS ini memiliki keamanan yang lebih baik dibandingkan dengan jalur internet umumnya.

Untuk melakukan proses enkripsi dan dekripsi pada OTP, diperlukan sebuah metode matematis yang dapat mengacak kode OTP tersebut. Metode matematis tersebut biasanya menggunakan algoritma khusus seperti halnya algoritma kriptografi. Ada banyak algoritma kriptografi yang biasa digunakan untuk otentikasi OTP, seperti penggunaan Algoritma *Data Encryption Standar* (DES), *Advanced Encyrption Standard* (AES), RSA dan algoritma lainnya. Dengan penambahan algoritma kriptografi, menimbulkan pengaruh yang signifikan untuk ketahanan dan keamanan dari serangan yang dilakukan oleh pihak yang tidak bertanggung jawab. Pada penelitian yang dilakukan (Nugroho, Judhie Putra, & Ramadhan, 2017) tentang penggunaan algoritma kriptografi AES-256 bits dalam meningkatkan ketahanan dan keamanan otentikasi, dilakukan uji *Avalanche Effect* dengan nilai sekitar 50%. Hal ini menunjukkan bahwa penambahan algoritma kriptografi mampu meningkatkan ketahanan dan keamanan dalam otentikasi.

Akan tetapi, tidak semua algoritma kriptografi bisa ditambahkan untuk meningkatkan keamanan dan ketahan. Salah satu faktor yang dapat mendukung ketahanan dan keamanan proses otentikasi adalah kecepatan dari sebuah algoritma kriptografi. Algoritma kriptografi yang memiliki kecepatan yang cukup baik salah satunya adalah algoritma RC6. Pada penelitian (KumarVerma & Kumar Singh,

2012), dilakukan penelitian terhadap *execution time* dari algoritma RC6, Twofish dan Rijndael. Hasil dari penelitian tersebut dapat dilihat pada tabel dan gambar berikut ini.



	16-Bytes	24-Bytes	32-Bytes
RC6	252.03	260.73	266.8
Twofish	297.83	308.73	318.79
AES	300.33	315.29	329.89

Gambar 1.1-1 Perbandingan *Execution Time* antara algoritma RC6, Twofish dan Rijndael (KumarVerma & Kumar Singh, 2012)

Dari gambar 1.1-2, didapatkan hasil bahwa *execution time* rata-rata dari algoritma RC6 dengan panjang kunci 16 bytes yaitu 252.03 *millisecond*, untuk panjang kunci 24 bytes yaitu 260.73 *millisecond*, dan untuk panjang kunci 32 bytes yaitu sebesar 266.8 *millisecond*. Dari data-data tersebut dapat disimpulkan bahwa algoritma RC6 memiliki rata-rata *execution time* yang lebih baik dibandingkan dengan algoritma Twofish dan Rijndael. Kemudian pada penelitian (Survey & Princy, 2015), algoritma RC6 memiliki keamanan yang cukup baik. Pada tahun 2015, Jindal dalam (Jindal & Singh, 2015) melakukan pengujian *Avalanche effect* terhadap finalis *Advanced Encryption Standard* dan didapatkan hasil bahwa algoritma RC6 memiliki nilai sebesar 49%. Selain itu pada penelitian *Randomness Test* yang dilakukan oleh Sulak dalam (Sulak, Douganaksoy, Ege, & Koçak, 2010) menunjukan bahwa keacakan chipertext dari algoritma RC6 muncul pada putaran ke 4. Hasil dari penelitian tersebut dapat dilihat dalam tabel

Rnds	Freq.	B.Freq.	Run	L.Run	Ap.En.	C.Sum1	C.Sum2	Serial
3	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
4	0.2504	0.2245	0.0371	0.1145	0.1413	0.0755	0.9803	0.1453
5	0.6958	0.7972	0.9787	0.1823	0.6092	0.9830	0.5004	0.3810

Tabel 1.1-1 Hasil *Randomness Test* pada penelitian Sulak (Sulak et al., 2010)

Hasil dari penelitian-penelitian tersebut menunjukkan bahwa, algoritma RC6 bisa digunakan untuk menambah nilai ketahanan dan keamanan pada proses otentikasi dan akan digunakan dalam penelitian ini.

1.2 Rumusan Masalah Penelitian

Berdasarkan uraian latar belakang yang telah dijelaskan, dapat dirumuskan
permasalahan pada penelitian ini adalah sebagai berikut:

1. Bagaimana mengimplementasikan Algoritma RC6 dan metode *One Time Password* pada pada proses pembuatan kode otentikasi untuk *website e-voting*?
2. Bagaimana hasil pengujian Algoritma RC6 dan metode *One Time Password* pada proses otentikasi untuk *website e-voting*?

1.3 Tujuan Penelitian

Tujuan dari penelitian ini adalah:

1. Mengimplementasi Algoritma RC6 dan metode *One Time Password* pada pada proses pembuatan kode otentikasi untuk *website e-voting*.
2. Memberikan hasil pengujian Algoritma RC6 dan metode *One Time Password* pada proses otentikasi untuk *website e-voting.*

1.4 Batasan Masalah

Batasan masalah pada penelitian ini adalah:

1. Penelitian dilakukan pada proses pengimplementasian RC6 pada proses otentikasi
2. Pembangkitan kode otentikasi pada saat proses masuk pengguna

1.5 Sistematika Penulisan

Sistematika penulisan proposal skripsi adalah sebagai berikut

BAB I PENDAHULUAN

Bab ini berisi latar belakang penelitian yang mencakup pengantar kode otentikasi dan yang menjadi landasan dalam pengembangan kasus perlindungan terhadap proses masuk pengguna ke aplikasi dalam penelitian sebelumnya yang akan menjadi landasan dalam penelitian ini. Selanjutnya bab ini berisi rumusan masalah penelitian, tujuan penelitian, batasan masalah, manfaat penelitian dan sistematika penulisan.

BAB II KAJIAN PUSTAKA

Bab ini berisi teori dan konsep terkait dalam penelitian yang menunjang pada penelitian ini

BAB III METODOLOGI

Bab ini berisi langkah-langkah penelitian yang diilustrasikan dengan skema desain penelitian, metode penelitian yang terdiri dari studi literatur dan proses pengembangan perangkat lunak, dan alat maupun bahan penelitian yang digunakan.

BAB II
TINJAUAN PUSTAKA

Bab ini akan menjelaskan mengenai teori-teori yang digunakan dan menunjang dalam melakukan penelitian.

2.1 **Kriptografi**

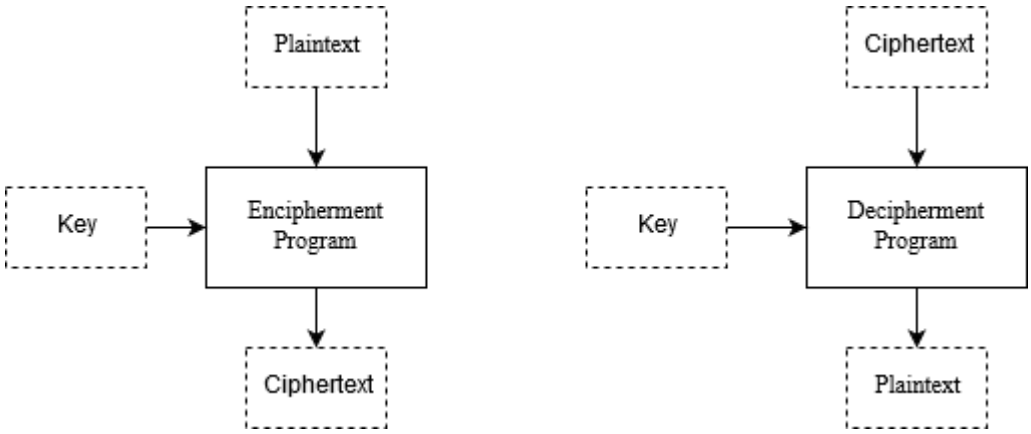
Kata kriptografi atau *"Cryptography"* berasal dari bahasa Yunani yaitu dari kata *kryptos* yang berarti tersembunyi dan kata *graphien* yang berarti tulisan. Para sejarawan percaya bahwa huruf hieroglif yang berasal dari Mesir sekitar tahun 1900 SM. Merupakan salah satu bentuk awal dari penyandian kata (Konheim, 2006).

Selain itu, kriptografi juga bisa dikatakan sebagai seni dan ilmu dalam menjaga keamanan pesan (Schneier, 2007).Secara umum, kriptografi adalah ilmu matematika terapan yang berhubungan dengan mentransformasikan data agar maksud dan makna dari data tersebut menjadi sulit untuk dipahami, agar data tersebut tidak dirubah tanpa izin atau data tersebut tidak digunakan secara sewenang-wenang. Selain itu, kriptografi juga diartikan sebagai suatu proses untuk mengembalikan data yang sudah terenkripsi menjadi bentuk yang dapat dipahami. Dalam mengamankan sebuah pesan, ada beberapa komponen yang terlibat yang harus diperhatikan antara lain (Schneier, 2007):

- Sender, merupakan pihak yang mengrimkan pesan.
- Receiver, merupakan pihak yang menerima pesan.
- Message atau *plaintext*, merupakan data atau informasi yang dapat dibaca, didengar, dilihat serta dimengerti makna dan isinya. Adapun pesan dapat berbentuk teks, gambar, suara, video serta data atau informasi lainnya.
- *Chipertext*, merupakan pesan atau *plaintext* yang telah dirubah menjadi acak dan tidak bisa terbaca
- Enkripsi, merupakan proses dimana merubah *plaintext* menjadi *chipertext*.

- Dekripsi, merupakan proses pengembalian pesan dari *chipertext* menjadi *plaintext* kembali.

Secara umum, proses enkripsi dan dekripsi dapat dilihat pada gambar dibawah ini.



Gambar 2.1-1 Proses Enkripsi dan Dekripsi. Sumber: (Konheim, 2006).

Secara matematis, proses dari enkripsi dan dekripsi dapat dinotasikan sebagai berikut (dimana E adalah Enkripsi, D adalah Dekripsi, C adalah *Chipertext*, M adalaha *Message*) (Schneier, 2007):

- Fungsi Enkripsi, dimana E memetakan M menjadi C

$$E\ (M) = C$$

- Fungsi Dekripsi, dimana D memetakan C menjadi M

$$D\ (C) = M$$

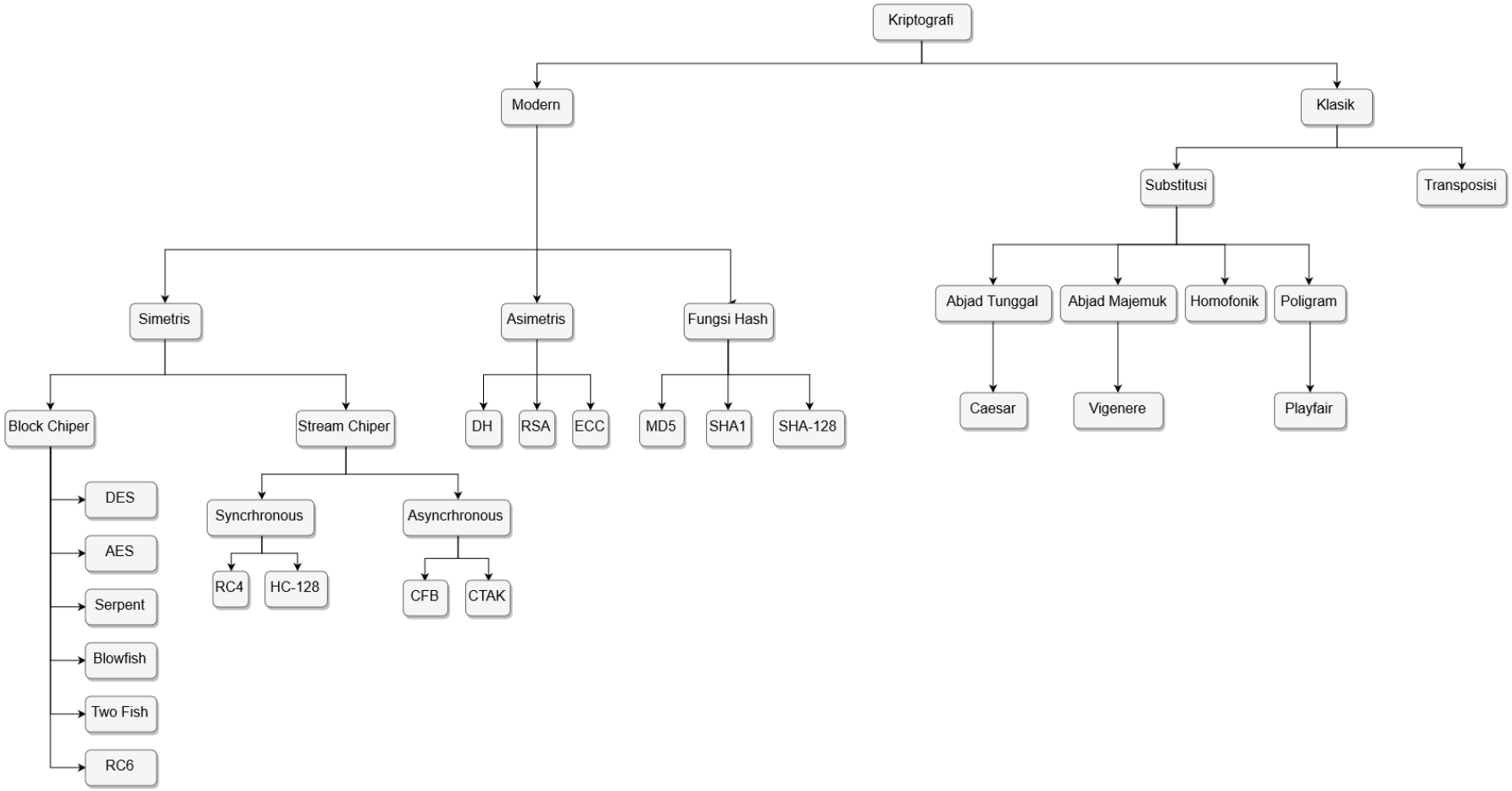
- Sehingga kita dapat menyimpulkan bahwa fungsi Dekripsi dan Enkripsi harus memenuhi sifat identitas sebagai berikut

$$D(E\ (M)) = M$$

Dalam proses enkripsi dan dekripsi dari sebuah pesan, (Munir, 2006) menjabarkan bahwa, ada 4 unsur utama yang harus dimiliki dari sebuah algoritma kriptografi, diantaranya adalah:

- Kerahasiaan (*Confidentiality)*, yaitu unsur yang digunakan untuk menjaga isi pesan dari siapapun yang tidak berhak untuk membacanya.
- Integritas Data (*Data Integrity)*, yaitu unsur yang menjamin bahwa pesan masih asli/utuh atau belum pernah dimanipulasi selama pengiriman.
- Otentikasi (*Authentication*), yaitu unsur yang digunakan untuk mengidentifikasi kebenaran pihak-pihak yang berkomunikasi (*user authentication*) dan untuk mengidentifikasi kebenaran sumber pesan (*data origin authentication*).
- Nir-penyangkalan (*Nonrepudiation*), yaitu unsur yang digunakan untuk mencegah entitas yang berkomunikasi melakukan penyangkalan, yaitu pengirim pesan menyangkal melakukan pengiriman atau penerima pesan menyangkal telah menerima pesan.

2.1.1 Jenis Algoritma Kriptografi



Gambar 2.1-2 Jenis Agoritma Kriptografi

2.1.1.1 Berdasarkan Jenis Kunci

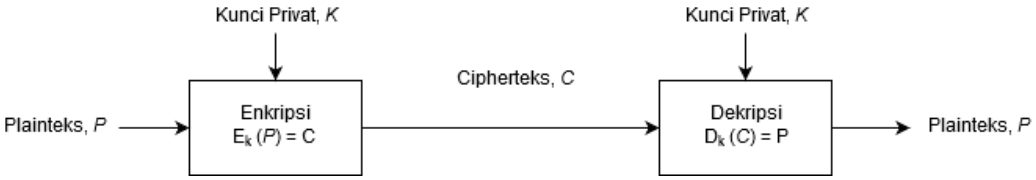
Berdasarkan jenis kuncinya, algoritma kriptografi dikelompokan menjadi dua bagian, yaitu: algoritma simetris (algoritma kunci privat) dan algoritma asimetris (algoritma kunci publik) (Menezes, Van Oorschot, & Vanstone, 1996).

1 Algoritma Simetris

Algoritma simetris adalah satu jenis kunci pada algoritma kriptografi yang menggunakan kunci enkripsi yang sama dengan kunci dekripsinya. Istilah lain untuk kriptografi kunci simetri adalah kriptografi kunci privat (*private-key cryptography*). Sistem kriptografi kunci simetri diasumsikan pengirim dan penerima pesan sudah berbagi kunci yang sama sebelum bertukar pesan. Keamanan sistem kriptografi simetri terletak pada kerahasiaan kuncinya.

Kriptografi simetri adalah jenis kriptografi yang diketahui masuk ke dalam catatan sejarah hingga tahun 1976. Semua algoritma kriptografi klasik termasuk ke dalam sistem kriptografi simetri. Salah satu kelebihan dari algoritma simetri yaitu proses enkripsi dan dekripsinya jauh lebih cepat dibandingkan dengan algoritma asimetris. Sedangkan kelamahannya yaitu pada permasalahan distribusi kunci (*key distribution*). Selain itu, karena proses enkripsi dan dekripsi pada kriptografi simetri menggunakan kunci yang sama. Sehingga timbul persoalan untuk menjaga kerahasiaan kunci. Maka dari itu, pengiriman *private key* harus melalui jalur yang berbeda dengan jalur umum (jalur pengiriman pesan), dengan menggunakan jalur khusus tersebut, maka dibutuhkan biaya yang tak sedikit. Kelemahan lain dari algoritma simetri lainnya adalah masalah efisiensi jumlah kunci. Jika terdapat *n* user, maka diperlukan *n(n-1)/2* kunci, sehingga untuk jumlah

user yang sangat banyak, sistem ini tidak efisien lagi (Menezes et al., 1996)



Gambar 2.1-3 Skema Kriptografi Simetri (Menezes et al., 1996)

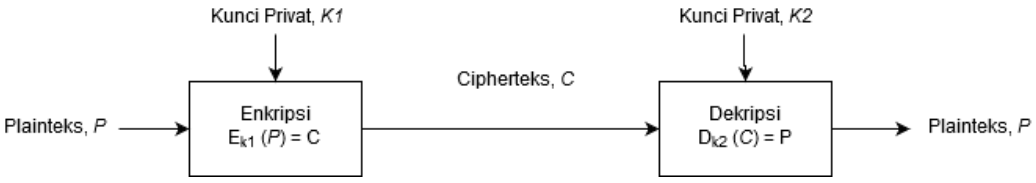
2 Algoritma Asimetri

Algoritma asimetris atau dapat disebut juga dengan algoritma kunci publik, didesain sebaik mungkin sehingga kunci yang digunakan untuk enkripsi berbeda dengan kunci dekripsinya. Dimana kunci untuk enkripsi tidak rahasia (diumumkan ke publik), sementara kunci dekripsinya bersifat rahasia (hanya diketahui oleh penerima pesan).

Pada kriptografi asimetris, setiap orang yang akan berkomunikasi harus mempunyai sepasang kunci, yaitu kunci privat dan kunci publik. Pengirim pesan akan mengenkripsi pesan menggunakan kunci publik si penerima pesan dan hanya penerima pesan yang dapat mendekripsi pesan tersebut karena hanya ia yang mengetahui kunci privatnya sendiri. Kriptografi kunci-publik dapat dianalogikan seperti kotak surat yang terkunci dan memiliki lubang untuk memasukan surat. Setiap orang dapat memasukkan surat ke dalam kotak surat tersebut, tetapi hanya pemilik kotak yang dapat membuka kotak dan membaca surat di dalamnya karena ia yang memiliki kunci. Sistem ini memiliki dau keuntungan. Yang pertama yaitu, tidak ada kebutuhan untuk mendistribusikan kunci privat sebagaimana pada sistem kriptografi simetri. Kunci publik dapat dikirim ke penerima pesan melalui saluran yang sama dengan saluran yang digunakan untuk mengirim pesan. Saluran untuk mengirim pesan umumnya tidak aman.

Kedua, jumlah kunci yang digunakan untuk berkomunikasi secara rahasia dengan banyak orang tidak perlu sebanyak jumlah orang tersebut, cukup membuat dua buah kunci, yaitu kunci publik bagi para pengirim pesan dan kunci privat untuk mendekripsi pesan. Berbeda dengan kriptografi kunci simetri yang membuat kunci sebanyak jumlah pihak yang diajak berkomunikasi.

Meski masih terbilang baru (sejak 1976), kriptografi kunci-publik mempunyai kontribus yang luar biasa dibandingkan dengan sistem kriptografi simetri. Kontribusi yang paling penting adalah tanda-tangan digital pada pesan untuk memberikan aspek keamanan otentikasi, integritas data, dan nirpenyangkalan. Tanda-tangan digital adalah nilai kriptografis yang bergantung pada isi pesan dan kunci yang digunakan. Pengirim pesan dapat mengenkripsi pesan (yang sudah diringkas) dengan kunci privatnya, hasil enkripsi inilah yang dinamakan tanda-tangan digital. Tanda-tangan digital dilekatkan (embed) pada pesan asli. Penerima pesan dapat memverifikasi tanda-tangan digital dengan menggunakan kunci publik.



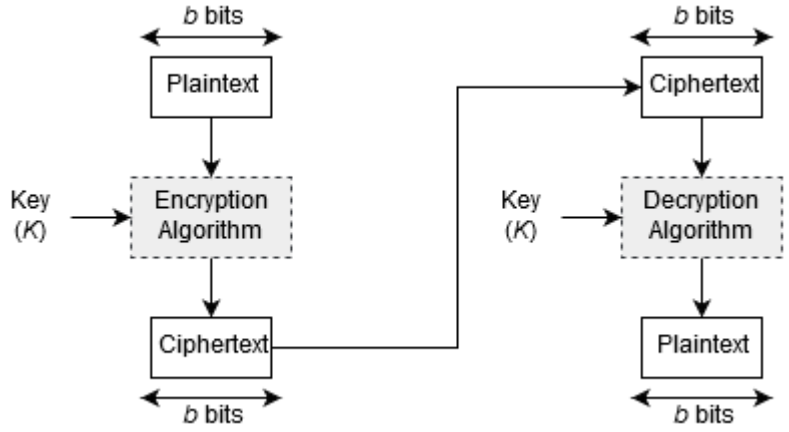
Gambar 2.1-4 Skema Kriptografi Asimetri (Menezes et al., 1996)

2.1.1.2 Berdasarkan Basis Bit

Berdasarkan basis bitnya, algoritma kriptografi dikelompokan menjadi dua bagian, yaitu: algortima cipher blok (*Block Chiper*) dan algoritma cipher aliran (*Stream Cipher*).

1. Cipher Blok (*Block Cipher*)

Cipher blok adalah blok bit dari plainteks yang diperlakukan secara keseluruhan dan digunakan untuk menghasilkan blok ciphertext dengan panjang yang sama. Biasanya, ukuran blok 64 atau 128 bit digunakan untuk melakukan enkripsi dan dekripsi.



Gambar 2.1-5 Skema enkripsi dan dekripsi cipher blok (Stallings, 2006)

Sebuah blok cipher beroperasi pada blok plaintext n bit untuk menghasilkan sebuah ciphertext dengan blok n bit yang sama jumlahnya. Ada 2^n blok plaintext yang berbeda dan, untuk enkripsi menjadi reversibel (yaitu, untuk dekripsi menjadi mungkin), masing-masing harus menghasilkan blok ciphertext yang unik.

Transformasi seperti itu disebut reversibel, atau nonsingular. Contoh berikut menggambarkan transformasi nonsingular dan singular untuk n = 2.

Reversible Mapping	
Plaintext	**Ciphertext**
00	11
01	10
10	00
11	01

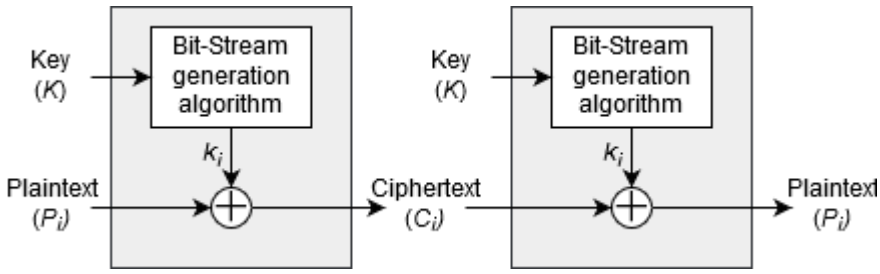
Irreversible Mapping	
Plaintext	**Ciphertext**
00	11
01	10
10	01
11	01

Tabel 2.1-1 Contoh transformasi *reversible* dan transformasi *irreversible* (Stallings, 2006)

Dalam kasus terakhir, sebuah ciphertext 01 bisa dihasilkan oleh salah satu dari dua blok plaintext. Jadi jika kita membatasi diri pada pemetaan reversibel, jumlah transformasi yang berbeda adalah 2^n.

2. Cipher Aliran (*Stream Cipher*)

Stream cipher adalah salah satu jenis algoritma kriptografi yang mengenkripsi aliran data digital satu bit atau satu byte dalam sekali waktu. Contoh ciphers aliran klasik adalah cipher Vigenère *autokeyed* dan cipher Vernam. Dalam kasus yang ideal, versi *one-time pad* dari Vernam cipher akan digunakan, di mana *keystream* (k_i) adalah sama panjangnya dengan panjang aliran bit plaintext (p_i). Jika *keystream* kriptografi acak, maka cipher ini tidak dapat dipecahkan dengan cara lain selain memperoleh *keystream*. Namun, kunci harus diberikan kepada kedua pengguna terlebih dahulu melalui beberapa saluran independen dan aman. Hal ini menimbulkan masalah logistik pengiriman kunci yang tidak dapat diatasi jika digunakan melalui pengiriman data melalui jalur umum. Oleh karena itu, untuk alasan praktis, generator bit-stream harus diimplementasikan sebagai prosedur algoritmik, sehingga aliran bit kriptografi bisa diproduksi oleh kedua pengguna. Dalam pendekatan ini, bit-stream Generator adalah algoritma *key-controlled* dan harus menghasilkan aliran bit yang kuat secara kriptografi. Artinya, harus tidak praktis memprediksi aliran bit yang baru berdasarkan bagian sebelumnya dari bit aliran. Kedua pengguna hanya perlu berbagi kunci pembangkit, dan masing-masing bisa menghasilkan *keystream*. Adapun skema enkripsi dan dekripsi dari cipher aliran dapat dilihat pada gambar dibawah ini.



Gambar 2.1-6 Skema Enkripsi dan dekripsi dari cipher aliran (Stallings, 2006)

Pada gambar diatas, proses enkripsi dan dekripsi dilakukan dengan meng-XOR-kan anatara plaintext (p_i) dengan kunci (k_i) untuk proses enkripsi dan meng-XOR-kan anatara ciphertext (c_i) dengan kunci (k_i) untuk proses dekripsi.

2.2 **Otentikasi**

Menurut Huang dalam penelitiannya (Huang, Huang, Zhao, & Lai, 2013), otentikasi adalah pemilik identitas "membuktikan" identitasnya kepada pemberi verifikasi dengan memberikan keunikannya atau ketepatan waktu untuk menjamin kerahasian data pribadi yang dimiliki. Sedangkan menurut Kshemkalyani dalam bukunya (Kshemkalyani & Singhal, 2011) mengatakan bahwa otentikasi merupakan serangkaian kegiatan identifikasi dan verifikasi untuk membuktikan kepemilikan dari si pemilik identitas.

2.2.1 Jenis-jenis Otentikasi

Ada beberapa teknik yang dapat membuktikan kepemilikan identitas atau otentikasi seperti dijelaskan pada peneiltian (Uymatiao & Yu, 2014) yaitu sebagai berikut.

a. Username dan password

Mayoritas website yang di gunakan saat ini menggunakan skema otentikasi tradisional yaitu dengan menggunakan *username* dan *password* melalui koneksi yang aman. *Username* digunakan untuk mengidentifikasi akun online mana yang ingin diakses oleh klien, sedangkan *password* digunakan untuk membuktikan identitas klien. Hal ini dianggap skema terbaik bahwa server hanya menyimpan hash

kriptografi dari *password*, dari pada menyimpan *password* sebagai plaintext. Dengan cara ini, data yang bocor tidak akan mengungkapkan informasi tentang *password* yang asli ketika penyerang menebak passwordnya dengan teknik *brute force.*

Meski pada prinsipnya terdengar cukup aman, pada kenyataannya banyak *password* berhasil diketahui. Hal ini bisa terjadi karena adanya keterhubungan dengan dua hal yakni *password* yang lemah dan juga cepatnya *cracking password* yang dilakukan oleh perangkat keras. Penggunaan *password* yang lemah dapat menghilangkan kekuatan pada hasil *hashing* karena *password* tersebut mudah ditebak, dengan kata lain pencocokan antara *password* yang telah di-*hashing* dengan *password* dalam bentuk plaintext akan lebih cepat (Gaw & Felten, 2006). Penggunaan *password* yang panjang dan juga kompleks dapat menyelesaikan masalah ini, akan tetapi *password* tersebut sangat susah untuk diingat. Dengan kemajuan perangkat keras juga membuat password lebih rentan untuk diketahui (Sprengers, 2011). Jika algoritma *hashing* yang dipilih oleh penyedia layanan tidak dirancang dengan sengaja menjadi lambat, maka penyerang akan bisa menebak password dengan kecepatan yang sama dengan kecepatan algoritma *hashing* tersebut, mengurangi jumlah waktu yang dibutuhkan pada proses *hashing* dapat membahayakan keamanan dari sebuah akun *online*. Berikut ini merupakan contoh dari *password* yang sering digunakan untuk otentikasi.

Tabel 2.2-1 Contoh *password* untuk otentikasi (Cheng, 2011)

Password Type	Example	Strength	Resist Brute Force or Dictionary attacks
Simple and basic	1234, dog	Poor	No
Long and random number	910482563957	Poor	No
Long phrase	ilikemyschool	Better	May be
Case sensitive	Steve, F1CarRace	Better	May be
Alpha-numeric	a12f5g, Jeff1234	Higher	Yes
Alpha-numeric with symbol	joe@itu168$	Higher	Yes

b. Multi factor

Selain dengan menggunakan *username* dan *password*, *c*ara yang paling populer untuk meningkatkan keamanan sebuah akun *online* adalah dengan meminta informasi tambahan dari pengguna. Selain meminta satu informasi dari user (seperti password), *server* bisa meminta informasi tambahan, sehingga dapat menyulitkan penyerang untuk memalsukan identitas pengguna. Pendekatan ini disebut autentikasi multi faktor, dan faktor tambahan ini biasanya bisa berbentuk sidik jari, iris mata, serta pertanyaan seputar pengguna yang bersifat rahasia atau hanya pengguna saja yang mengetahuinya, hal tersebut dilakukan untuk mengetahui bahwa pengguna memiliki sesuatu khusus yang bisa dikenali oleh server selain password.

c. One-time Password (OTP)

One time Password sebagai teknik otentikasi pertama kali dikenalkan oleh Leslie Lamport (Lamport, 1981). *One Time Password* dikenal juga sebagai enkripsi *Verman* atau algoritma enkripsi sempurna yang mana *Plaintext* digabungkan dengan kunci acak. Teknik itu merupakan salah satu metode yang diketahui berasal dari enkripsi matematika yang tidak bisa diselesaikan. OTP merupakan algoritma simetris yang menggunakan kunci yang sama untuk proses enkripsi dan dekripsi. Proses enkripsi dan dekripsi pada teknik ini biasanya menggunakan operasi XOR untuk men-*generate* berkas enkripsi. Bruce Scheiner dalam (Schneier, 2007) mengatakan bahwa OTP merupakan skema enkripsi yang sempurna dengan kondisi jika kuncinya benar-benar acak dan tidak pernah digunakan lagi.

Metode ini memberikan garansi keamanan yang baik, meskipun penyerang mendapatkan password pada jaringan internet atau dicuri dari pengguna. Selain itu, OTP memiliki beberapa fitur seperti kerahasiaan, mudah dibawa, dan memiliki cakupan yang sangat luas (Cho et al., 2009).

Sistem OTP bergantung pada kemampuan sebuah perangkat untuk menghasilkan kode yang dapat digunakan satu kali dan akan dikirim ke server untuk dilakukan verifikasi. Jika kode yang diverifikas benar, maka pengguna tersebut diberi akses ke akunnya. Generasi OTP ini melibatkan setidaknya tiga hal yakni algoritma generasi OTP, *secret seed*, dan *counter* untuk menghitung berapakali kode tersebut digunakn. Ketiga hal tersebut bekerja sama sebagai sebuah fungsi:

$$password = OTP(seed, counter)$$

Saat fungsi tersebut dipanggil, nilai dari *counter* akan bertambah, artinya *output* kode untuk verifikasi akan selalu berbeda setelah setiap fungsi dipanggil. Hal ini juga penting, bahwa untuk setiap *output* yang diberikan dari algoritma OTP, *output* itu tidak boleh mengungkapkan informasi apapun tentang *seed* atau output yang akan dihasilkan nantinya. Bila diimplementasikan dengan benar, *one time password* bisa dianggap aman karena kode tersebut hanya bisa digunakan satu kali. Bahkan jika penyerang berhasil mendapatkan kode ini, kode tersebut tidak dapat digunakan untuk menerobos akun korban.

Menurut Lee (Lee, Lim, & Lee, 2010) ada beberapa cara yang bisa digunakan untuk membangkitkan kode OTP, diantaranya sebagai berikut.

1) OTP bertipe *Challenge-Response*.

Sistem generator OTP menggunakan *password* dari pengguna dan digabungkan dengan sebuah *seed* yang diterima dari server sebagai bagian dari tantangan (masukkan). Kemudian, *password* dan *seed* akan melalui beberapa iterasi dari fungsi *hash* yang nantinya akan menghasilkan sebuah kode otentikasi yang dapat digunakan satu kali saja. Setelah kode tersebut berhasil terotentikasi ataupun telah dibangkitkan, jumlah pembangkitan kode OTP dengan fungsi *hash* akan berkurang sebanyak satu kali. Dengan demikian, kode *password* yang unik telah ter-*generate* secara berurutan. Server memverifikasi kode *password* tersebut

yang diterima dari generator dengan menggunakan fungsi *hash* dalam satu iterasi dan membandingkan hasilnya dengan kode *password* yang diterima sebelumnya.

2) OTP bertipe *Time-Synchronized*

OTP bertipe *Time-Synchronized* biasanya berhubungan dengan fisik dari perangkat keras yang meruquest token (mis., setiap pengguna diberi token/kode yang digunakan untuk verifikasi OTP). Di dalam token ini, ada sebuah jam untuk menghitung waktu secara akurat dan telah disinkronisasi dengan jam aktif pada server otentikasi. Pada sistem OTP ini, waktu adalah bagian terpenting dari algoritma password sejak generasi baru password didasarkan pada waktu sekarang (waktu terjadinya request kode OTP) dari pada password sebelumnya atau kunci rahasia. Secara sederhananya, prinsip kerja dari OTP bertipe *Time-Synchronized* adalah ketika user melakukan otentikasi dan meminta kode otentikasi dari server, maka secara otomatis server akan mengirimkan kode tersebut kepada pengguna berserta watu yang nantinya akan tersinkronisasi dengan waktu server. Ketika pengguna melakukuan otentikasi dengan kode yang telah dikirimkan sebelumnya, server akan mengecek kode otentikasi dan waktu sinkronisasinya. Jika kode otentikasi yang dimasukkan oleh pengguna benar dan waktu sinkronisasi masih diambang batas waktu penginputan, maka otentikasi berhasil. Akan tetapi, jika kode otentikasi yang dimasukkan salah atau waktu sinkronisasi telah melebihi ambang batas waktu penginputan, maka server akan secara otomatis akan mengirimkan kode otentikasi baru dan juga akan mengatur ulang waktu sinkronisasi.

3) OTP bertipe *Event-Synchronized*

Setiap kali pengguna meminta password baru pada sistem OTP bertipe *Event-Synchronized*, maka nilai *counter* pada sistem akan bertambah secara otomatis sebanyak satu. Di server, setiap kali pengguna berhasil melakukan otentikasi, server juga akan menambahkan nilai *counter*-nya

sebanyak satu. Dengan cara ini, nilai *counter* token dan server tetap tersinkronisasi dalam langkah yang sama dan akan selalu menghasilkan kode OTP yang sama. Sinkroniasasi dari token yang berbasis *event* dapat berhenti jika token diminta untuk menghasilkan sekumpulan kode otentikasi OTP yang tidak pernah digunakan untuk otentikasi. Kemudian, nilai *counter* pada token akan bertambah, sedangkan nilai *counter* pada server tidak bertambah. Akhirnya, token yang dihasilkan dari OTP digunakan untuk otentikasi akan gagal karena server tidak mengenalinya.

4) OTP bertipe *Time-Event-Synchronized*

Dengan penggabungan kedua jenis OTP antara token berbasis *event* dan token berbasis waktu, memungkinkan server untuk melakukan perbaikan secara otomatis untuk mengatasi masalah sinkronisasi, dengan batasas/kondisi tertentu. Untuk token yang berbasis *event*, server selalu tahu nilai terakhir dari nilai counter saat ini (nilai counter yang digunakan untuk otentikasi yang berhasil sebelumnya) akan tetapi bukan batas atas dari nilai *counter* berikutnya. Karena itu, jika kode OTP yang tidak dikenali terlihat, server bisa mencoba beberapa kode OTP diluar dari nilai *counter* yang saat ini untuk melihat apakah ada yang cocok atau tidak. Jika ada yang cocok, maka server tahu bahwa adanya campur tangan/gangguan terhadap nilai *counter* yang telah "hilang" dan harus melewati nilai *counter* tersebut. Untuk token berbasis waktu, strategi serupa diterapkan pada beberapa interval waktu baik itu waktu sebelumnya maupun waktu yang akan datang untuk menyinkronkan secara otomatis waktu pada server dengan waktu pada perangkat pengguna. Tentu saja, penggunaan token secara umum diperlukan untuk menjaga sinkronisasi waktu antara server dan perangkat pengguna.

Meskipun ada banyak kemungkinan penerapan OTP, ada dua algoritma yang didukung oleh *Initiative for Open Authentication* (OATH) untuk digunakan yakni HOTP (*One-time Password* berbasiskan HMAC) (M’raihi, Bellare, Hoornaert, Naccache, & Ranen, 2005) dan juga

pengimbangnya yang berbasiskan waktu yakni TOTP (*One-time Password* berbasiskan waktu) (M'Raihi, Machani, Pei, & Rydell, 2011). Output dari algoritma ini adalah angka numerik dengan panjang 6 digit.

Selain itu, ada beberapa cara yang bisa dilakukan untuk mengirimkan kode otentikasi OTP, yaitu sebagai berikut.

1) SMS OTP

Salah satu cara untuk mengirimkan kode OTP kepada pengguna adalah melalui SMS OTP. Pengguna diharuskan untuk mengisi *username* dan *password* pada saat proses *login.* Kemudian, *username* dan *password* tersebut akan dikirmkan ke *server* untuk dienkripsi dan menghasilkan kode OTP. *Server* menghasilkan kode OTP menggunakan algoritma kriptografi tertentu. Setelah proses enkripsi selasai, kode OTP dikirimkan kembali kepada pengguna melalui layanan SMS. Pengguna diharuskan mengisi kode OTP yang tertera pada SMS ke aplikasi untuk nantinya dikirimkan kembali ke *server*. Otentikasi terjadi pada saat server mengenali bahwa pengguna memasukkan kode yang benar saat login. Setelah kode OTP diterima oleh *server*, kode tersebut didekripsi dan kemudian dicocokan dengan kode OTP yang belum dienkripsi sama sekali. Dengan menggunakan layanan SMS ini dapat mengurangi biaya dari penggunaan perangkat keras tambahan. Selain itu, pada penelitian yang dilakukan (Wu et al., 2004) menunjukkan bahwa layanan SMS ini memiliki keamanan yang lebih baik dibandingkan dengan jalur internet umumnya.

2) Web Authenticator

Dalam penelitian terbaru, beberapa penelitian menggunakan *web* untuk mengirimkan kode perantara/token untuk mengarahkan pengguna melakukan otentikasi dengan kode OTP yang sebenarnya. Pengiriman kode perantara/token biasanya melalui saluran internet khusus seperti pengiriman e-mail atau pengguna diarahkan ke halaman lain. Sayangnya,

pengiriman semacam ini membutuhkan jalur khusus seperti penggunaan SSL ataupun HTTPS dan hal itu membutuhkan biaya yang cukup besar.

2.3 Algoritma RC6

2.3.1 Sejarah Algoritma RC6

RC6 adalah algoritma cipher blok baru yang diusulkan kepada NIST oleh Ronald L. Rivest dalam penelitiannya (Rivest, Robshaw, Sidney, & Yin, 1998) untuk dipertimbangkan sebagai Standar Algoritma Enkripsi (AES) terbaru. Desain RC6 merupakan pengembangan dari RC5 sebagai salah satu calon potensial yang diusulkan untuk AES. Modifikasi algoritma RC5 dilakukan untuk memenuhi Persyaratan AES, serta untuk meningkatkan keamanan, dan untuk meningkatkan kinerja algoritma RC5. Bagian *loop* dari algoritma RC6 didasarkan/sama dengan "*half-round*" yang ada pada algoritma RC5.

Secara praktek, tidak ada serangan yang ditemukan untuk memecahkan algoritma RC5. Berbagai penelitian memberikan teori yang menarik terhadap serangan yang dilakukan pada algoritma RC5, umumnya didasarkan pada fakta bahwa "jumlah rotasi" dari algoritma RC5 tidak bergantung kepada semua bit dalam register RC5 itu sendiri. RC6 dirancang untuk menggagalkan serangan tersebut, dan juga untuk menggagalkan semua serangan yang telah diketahui. Selain itu juga, dapat menyediakan sebuah algoritma enkripsi yang dapat memenuhi kebutuhan dari keamanan dan jangka waktu penggunaan AES.

Untuk memenuhi persyaratan AES tersebut, sebuah algoritma cipher blok dapat menangani 128-bit dari blok masukan ataupun blok keluaran. Sementara itu, RC5 adalah cipher blok yang sangat cepat, modifikasi yang paling umum dan normal yang bisa dilakukan pada algoritma RC5 agar bisa memproses 128-bit dari blok masukan dan blok keluaran adalah dengan mengguanakan 2 buah register 64-bit. Arsitektur dan bahasa target yang ditentukan untuk AES belum mendukung operasi 64-bit dengan efisien dan bersih. Dengan demikian, desain dari modifikasi

algoritma RC5 menggunakan empat buah 32-bit register dibandingkan dengan menggunakan 2 buah 64-bit register. Hal ini memberikan manfaat untuk melakukan putaran sebanyak dua kali dalam satu kali epoh dibandingkan dengan satu kali putaran dari *half-round* algoritma RC5, dan pada algoritma RC6 ini menggunakan banyak bit data untuk menghitung jumlah putaran setiap epohnya.

2.3.2 Detail Algoritma RC6

Algoritma RC6 merupakan bagian dari algoritma enkripsi dengan menggunakan parameter. Secara akurat dan spesifik, algoritma RC6 dapat ditulis seperti RC6-*w/r/b* dimana simbol *w* merupakan jumlah bit dari plaintext, *r* merupakan bilangan bulat positif dari jumlah epoh proses enkripsi dan *b* merupakan representasi dari kunci enkripsi dengan satuan *byte.* Ketika sebuah plaintext memiliki nilai *w* dan *r* yang berbeda dari standar untuk RC6 untuk AES, maka secara spesifik versi dari RC6 akan berubah menjadi RC6-*w/r*. Untuk semua varian dari algoritma RC6, ada beberapa operasi matematika terhadap ke empat buah *w*-bit dari plaintext dengan menggunakan operasi sebagai berikut. Bilangan biner dari *w* dilambangkan dengan lg *w*.

$a + b$ integer addition modulo 2^w

$a - b$ integer substraction modulo 2^w

$a \oplus b$ bitwise exclusive-or of *w*-bit words

$a \times b$ integer multiplication modulo 2^w

$a \lll b$ rotate the w-bit word a to the left by the amount given by the least signicant lg w bits of b

$a \ggg b$ rotate the w-bit word a to the right by the amount given by the least signicant lg w bits of b

2.3.3 Key Schedule

Key Schedule dari RC6 secara prakteknya mirip dengan *key schedule* dari RC5. Perbedaannya bereada pada jumlah kata dari kunci yang dimasukkan oleh pengguna selama proses enkripsi dan dekripsi dari

algoritma RC6. Pengguna memasukkan sejumlah *b* bytedari kunci. Sejumlah byte nol yang sesuai ditambahkan untuk memberikan panjang kunci yang sama terhadap terhadap jumlah bilangan dari kata yang bukan nol. Kemudian, byte-byte dari kunci tersebut disimpan disebuah array c dari *w*-bit plaintext (*L*[0],..... ,*L*[c-1]). Kemudian, byte pertama dari kunci disimpan kedalam urutan terendah dari sebuah array *L*[0] hingga seterusnya. Kemudian, array *L*[*c*-1] ditambahkan setelah urutan array tertinggi jika diperlukan. Jumlah *w*-bit plaintext yang akan dihasilkan dan ditambahkan yaitu sebanyak 2*r*+4 dan akan disimpan kedalam array S[0,...., 2*r*+4].

Konstanta P_{32} = B7E15163 dan Q_{32} = 9E3779B9 (heksadesimal) disebut juga sebagi "konstanta sihir" yang sama seperti yang digunakan dalam *key schedule* dari algoritma RC5. Nilai P_{32} berasal dari ekspansi biner dari *e*-2, di mana *e* adalah basis alami dari fungsi logaritma. Nilai Q_{32} berasal dari ekspansi biner ϕ - 1, dimana ϕ adalah Rasio Emas. Definisi serupa berasal dari RC5 untuk P_{64} seterusnya dapat digunakan untuk versi RC6 dengan ukuran plaintext yang berbeda. Nilai ini bisa berubah sewaktu-waktu, dan nilai lainnya dapat dipilih untuk memberikan ciri khas atau kepemilikan dari versi RC6. Secara ringkas, *key schedule* dari RC6 dapat dilihat pada gambar berikut.

Key Scedule for RC6-*w/r/b*

Input: User-supplied *b* byte key preloaded into the *c*-word array $L[0,...,c-1]$
Number *r* of rounds

Output: *w*-bit round keys $S[0,....,2r + 3]$

Procedure: $S[0] = P_w$

for $i = 1$ **to** $2r + 3$ **do**
$S[i] = S[i - 1] + \boldsymbol{Q_w}$

$A = B = i = j = 0$

$v = 3 \times \max\{c, 2r + 4\}$
for $s = 1$ **to** v **do**
{
$A = S[i] = (S[i] + A + B) <<< 3$
$B = L[j] = (L[j] + A + B) <<< (A + B)$
$i = (i + 1) \bmod (2r + 4)$
$j = (j + 1) \bmod c$
}

Gambar 2.3-1 *Psedeucode Key Schedule* algoritma RC6 (Rivest et al., 1998)

2.3.4 Enkripsi Dan Dekripsi

RC6 bekerja dengan empat register *w*-bit yaitu A, B, C, D yang berisi input awal plaintext serta output dari ciphertext pada akhir enkripsi. Byte pertama dari plaintext atau ciphertext ditempatkan dalam byte *least-significant* dari A. Kemudian, byte terakhir dari plaintext atau ciphertext ditempatkan ke dalam byte *most-significant* dari D. Kita menggunakan (A, B, C, D) = (B, C, D, A) yang berarti melakukan pergeseran nilai register dari sebelah kanan ke sebelah kiri secara paralel. Ada pun proses enkripsi dari algoritma RC6 dapat dilihat pada gambar berikut.

Encryption with RC6-*w/r/b*

Input: Plaintext stored in four w-bit input registers, A, B, C, D
Number r of rounds
w-bit round keys $S[0,....,2r + 3]$

Output: Ciphertext store in A, B, C, D

Procedure:
$B = B + S[0]$
$D = D + S[1]$
for $i = 1$ **to** ***r*** **do**

{

$t = (B \times (2B + 1)) <<< \lg w$
$u = (D \times (2D + 1)) <<< \lg w$
$A = ((A \oplus t) <<< u) + S[2i]$
$C = ((C \oplus u) <<< t) + S[2i + 1]$
$(A, B, C, D) = (B, C, D, A)$

}

$A = A + S[2r + 2]$
$C = C + S[2r + 3]$

Gambar 2.3-2 *Pseudocode* enkripsi algoritma RC6 (Rivest et al., 1998)

Sedangkan untuk proses dekripsi dari algoritma RC6 dapat dilihat pada gambar berikut.

Decryption with RC6-*w/r/b*

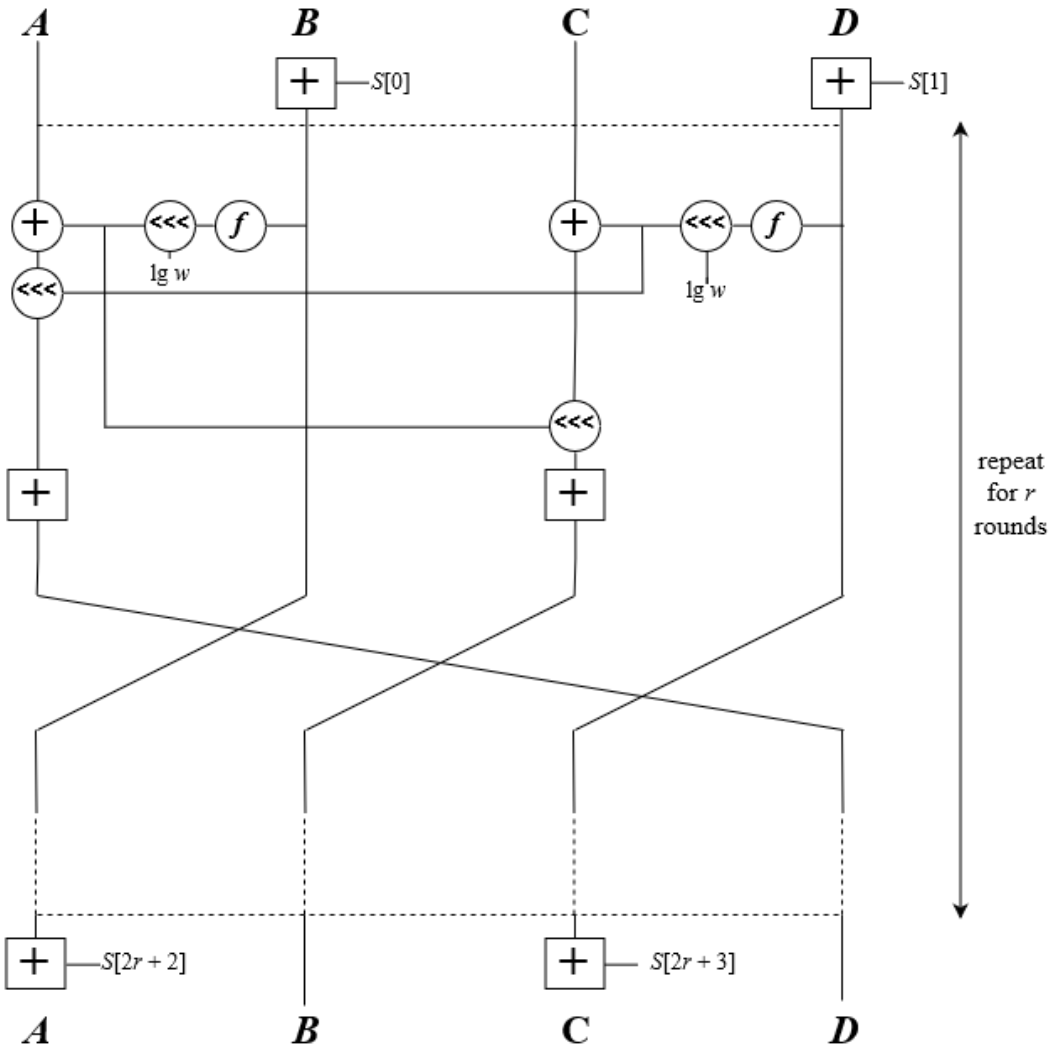
Input: Ciphertext stored in four *w*-bit input registers, *A*, *B*, *C*, *D*
Number *r* of rounds
w-bit round keys *S*[0,....,2*r* + 3]

Output: Plaintext store in *A*, *B*, *C*, *D*

Procedure:
C= *C* + *S*[2*r* + 3]
A= *A* + *S*[2*r* + 2]
for *i* = *r* down**to** 1 do
{
(*A*, *B*, *C*, *D*) = (*B*, *C*, *D*, *A*)
u = *(D* x *(2D + 1))* <<< lg *w*
t = (*B* x (2*B* + 1)) <<< lg *w*
C = ((*C* - *S*[2*i* + 1]) >>> *t*) ⊕ *u*
A = ((*A* - *S*[2*i*]) >>> *u*) ⊕ *t*
}
D = *D* - *S*[1]
B = *B* - *S*[0]

Gambar 2.3-3 *Pseudocode* dekripsi algoritma RC6 (Rivest et al., 1998)

Dari proses enkripsi dan dekripsi tersebut dapat dapat digabungkan menjadi sebuah diagram sebagai berikut



Gambar 2.3-4 Diagram algoritma RC6 (Rivest et al., 1998)

2.4 **Avalanche Effect**

Karakteristik yang dapat menentukan baik atau tidaknya suatu algoritma dalam kriptografi dapat dilihat pada *Avalanche Effect*-nya. Perubahan satu bit pada plainteks atau kunci dapat menyebabkan perubahan terhadap hasil dari cipherteks setidaknya setengah dari bit cipherteks (Kumar & Tiwari, 2012). Suatu *Avalanche Effect* dikatakan baik jika perubahan bit yang dihasilkan berkisar antara 45%-60% (50% merupakan hasil yang sangat baik). Hal ini disebabkan karena dengan adanya perubahan tersebut, maka

dapat memberikan perbedaan yang cukup sulit bagi kriptanalis dalam melakukan serangan. Nilai *Avalanche Effect* dirumuskan sebagai berikut:

$$Avalnche\ Effect\ (AE) = \frac{\Sigma bit\ berubah}{\Sigma bit\ total} \text{x } 100\%$$

2.5 **Randomness Test**

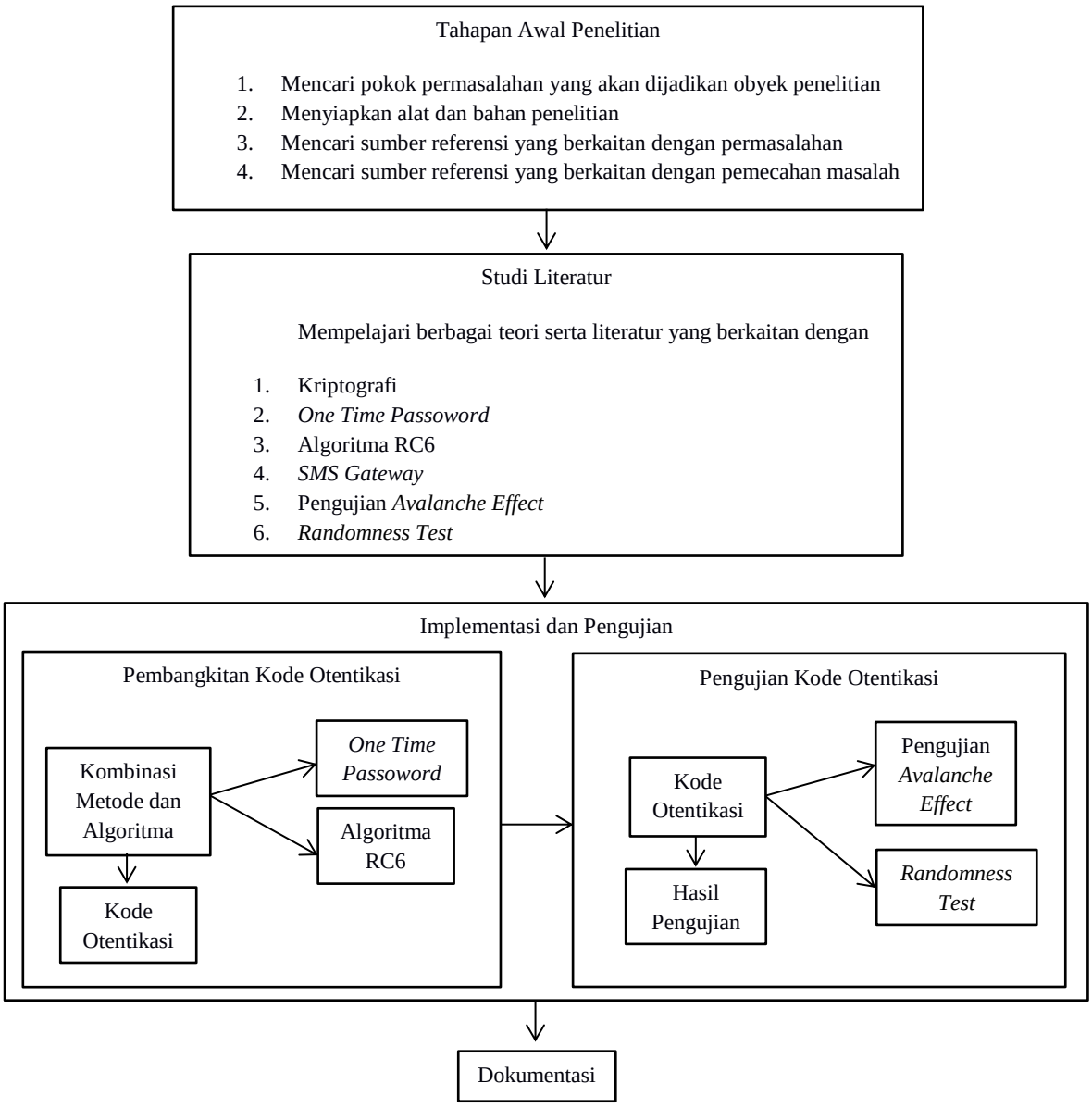
Randomness test merupakan uji deret untuk melihat keacakan. Tujuan dari uji deret untuk menentukan apakah dalam suatu data terdapat pola tertentu atau apakah data tersebut merupakan *sample* yang acak. Dari pelaksanaan *randomness test* ini akan didapat 2 kemungkinan, yaitu jika data bersifat acak (*random*) terhadap median maka data tersebut homogen, sedangkan jika data memiliki kecenderungan (*trend*) lebih banyak di atas median atau di bawah median maka data tersebut tidak homogen (Ramanujam & Karuppiah, 2011).

BAB III
METODE PENELITIAN

Bab ini membahas kerangka penelitian baik itu desain penelitian, metode penelitian serta alat dan bahan yang akan dilakukan selama penelitian berlangsung.

3.1 Desain Penelitian

Dalam melakukan penelitian ini, penulis menyusun sebuah kerangka kerja dalam bentuk diagram alur secara terurut dan tersusun, guna mempermudah dalam melakukan penelitian. Kerangka kerja atau desain penlitian ini dapat dilihat pada gambar 3.1



Gambar 3.1-1 Desain Penelitian

Penjelasan gambar 3.1-1 akan dijabarkan sebagai berikut

1. Tahap Awal Penelitian

Tahapan ini merupakan tahapan persiapan dari sebuah penelitian. Dimana dalam tahapan ini terdapat sebuah permasalahan yang nantinya akan menjadi tujuan dari penelitian ini. Selain itu, dalam tahapan ini pula perlu dipersiapkan alat dan bahan serta referensi dari permasalahan maupun metode pemecahan dari permasalahan itu sendiri yang nantinya akan menunjang dalam penelitian ini.

2. Studi Literatur

Studi literatur merupakan sebuah tahapan dalam penelitian untuk mendapatkan pemahaman dari literatur-literatur yang telah dikumpulkan pada tahapan sebelumnya baik itu berupa teori maupun pembahasan materi yang berkaitan dengan penelitian ini. Adapun literatur-literatur yaitu berkaitan dengan Kriptografi, *One Time Passoword*, Algoritma RC6, *SMS Gateway*, Pengujian *Avalanche Effect* dan *Randomness Test*. Sumber literatur pada penelitian ini berupa jurnal, teks buku, *website*, dan berbagai sumber lainnya. Selain itu pada tahapan ini pula, dilakukan proses latihan dari alat dan bahan guna mendukung proses penelitian ini.

3. Tahapan Implementasi dan Pengujian

Tahapan implementasi dan pengujian merupakan sebuah tahap inti dari penelitian yang dilakukan, dimana pada penelitian ini dilakukan sebuah implementasi gabungan antara metode *One Time Passoword* dengan Algoritma RC6 untuk mendapatkan sebuah kode otentikasi yang bersifat rahasia dan memiliki integritas data yang cukup baik dalam bentuk aplikasi dengan menggunakan bahasa pemrograman PHP. Setelah proses implementasi selesai, proses berikutnya adalah pengujian terhadap kode otentikasi yang dihasilkan. Adapun proses pengujian yang dilakukan antara lain pengujian *Avalanche Effect* dan *Randomness Test*. Tujuan dari

proses pengujian ini untuk mengetahui seberapa kuatkah kode otentikasi yang dihasilkan terhadap serangan yang dilakukan pada kode tersebut. Serta untuk mengetahui tingkat keacakan antara *plaintext* dengan *chipertext* yang telah dihasilkan.

4. Tahapan Dokumentasi

Tahapan dokumentasi merupakan sebuah tahapan yang bertujuan untuk mendokumentasikan seluruh kegiatan dan hasil dari penelitian ini.

3.2 Metode Penelitian

Metode penelitian yang digunakan dalam penelitian ini terbagi menjadi dua bagian yaitu metode yang dilakukan dalam mengumpulkan data serta metode yang dilakukan dalam proses pembuatan aplikasi.

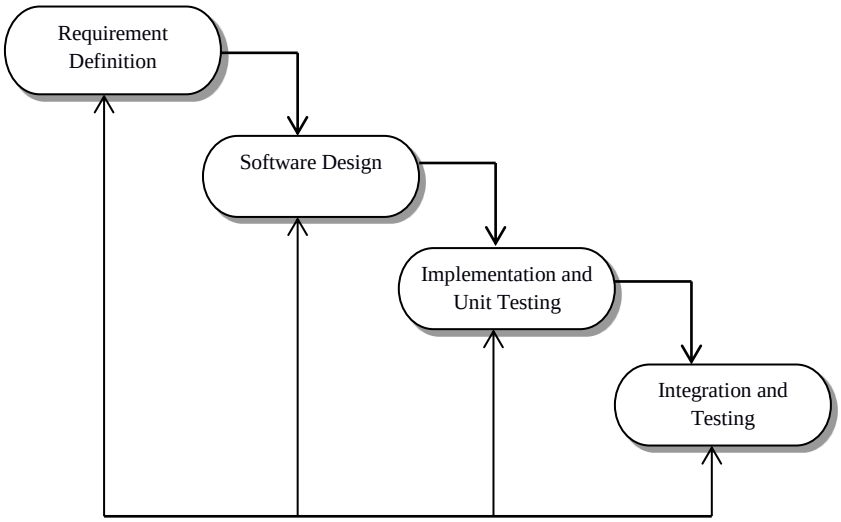
3.2.1 Metode Pengumpulan Data

Proses pengumpulan data dari penelitian ini menggunakan metode studi literatur. Yakni, sebuah metode yang dilakukan dengan mempelajari dan memahami teori dan pembahasan mengenai penelitian ini. Adapaun teori dan pembahasan yang dikumpulkan pada penelitian ini antara lain Kriptografi, *One Time Passoword*, Algoritma RC6, *SMS Gateway*, Pengujian *Avalanche Effect* dan *Randomness Test*. Sumber yang digunakan untuk mengumpulkan literatur-literatur tersebut berasal dari jurnal, buku teks, *website* serta sumber pendukung lainnya.

3.2.2 Metode Pengembangan Perangkat Lunak

Dalam proses pengembangan perangkat lunak dari penelitian ini, digunakan sebuah pendekatan yang digagas oleh Ian Sommerville (Sommerville, 2004) yaitu metode *Waterfall*. Metode tersebut dipilih karena sangat mudah diterapakan dalam pembangunan perangkat lunak dan memudahkan bagi pembuat perangkat lunak untuk melukan pengujian serta mendesain ulang apabila terdapat kesalahan dalam proses pembuatan perangkat lunak tersebut. Berikut ini merupakan metode

waterfall yang diterapkan pada penelitian ini yang akan ditunjukan oleh gambar 3.2-1



Gambar 3.2-1 Diagram *Waterfall* (Sommerville, 2004)

a. Requirement Definition

Dalam tahap ini, dilakukan pengumpulan informasi mengenai pembuatan aplikasi Otentikasi baik itu berupa jenis entitas, data, kebutuhan fungsional, kebutuhan non-fungsional serta batasan-batasan apa saja dalam membangun aplikasi otentikasi. Informasi-informasi tersebut didapatkan dari hasil studi literatur yang dilakukan pada tahapan sebelumnya. Tahapan ini dilakukan agar aplikasi yang dirancang memenuhi spesifikasi yang diinginkan sesuai dengan kebutuhan.

b. *Software* Design

Setelah memetakan kebutuhan dan spefisikasi yang akan dibuat, tahapan berikutnya adalah merancang sistem dari aplikasi otentikasi. Tahapan ini dilakukan untuk memudahkan penerjemahan dari bahasa manusia (kebutuhan dan spesifikasi perangkat lunak) kedalam bahasa program. Adapun alat atau metode yang digunakan dalam memodelkan dan merancang sistem otentikasi ini yaitu dengan

menggunakan diagram *Unified Modeling Languange* (UML). Pemilihan UML sebagai alat bantu untuk memodelkan kebutuhan perangkat lunak didasarkan kepada penggunaan *framework* dari bahasa pemrograman PHP yaitu *Code Igniter* yang bersifat pemrograman berbasis objek.

c. Implementation and Unit Testing

Setelah tahapan pemodelan selesai, langkah berikutnya adalah melakukan implementasi kedalam bahasa pemrograma PHP. Pemilihan bahasa tersebut dikarenakan bahasa PHP merupakan satu bahasa *server side* dalam membangkun aplikasi *website*. Dalam proses implentasi antara metode *One Time Passoword* dengan Algoritma RC6 tidak dibuat dari awal akan tetepi menggunakan salah satu *framework* dari bahasa pemrograman PHP yaitu *Code Igniter*. Pemilihan *framework* tersebut didasarkan kepada kemudahan dalam pembangunan aplikasi serta lengkapnya dokumentasi yang dimiliki oleh *Code Igniter*. Selain pembangunan aplikasi, untuk menghindari kesalahan dari sistem, maka pada proses pembangunan aplikasi ini dilakukan *Unit Testing* disetiap modul yang dibuat.

d. Integration and Testing

Setelah memastikan tidak ada kesalahan pada setiap modulnya, langkah berikutnya adalah menyatukan seluruh modul dan melakukan pengujian kembali, baik itu alur dari aplikasi maupun sistem secara keseluruhan. Jika dirasa sudah tidak ada kesalahan lagi, maka proses berikutnya adalah melakukan pengujian *Avalanche Effect* dan *Randomness Test*.

3.3 Alat dan Bahan

Dalam melakukan proses penelitian ini, penulis menggunakan berbagai alat dan bahan guna menunjang proses penelitian baik itu perangkat lunak, perangkat keras, serta bahan materi dan teori.

3.3.1 Alat Penelitian

Adapun perangkat keras (*hardware*) yang digunakan penulis, memilki spesifikasi sebagai berikut:

- *Processor* Intel i5-5200U 2.2 GHz
- RAM 8 GB
- *Harddisk Drive* 500GB
- *Mouse* dan *Keyboard*
- Layar *Monitor* 14 inch dengan resolusi 1366 x 768

Sementara, untuk perangkat lunak (*software*) yang digunakan adalah sebagai berikut:

- Text Editor Visual Studio Code versi 1.26.1 (x64 bit)
- WAMP Server 3.1.0 (x64 bit)
- PHP versi 7.1.9
- Apache versi 2.4.27
- Mysql versi 5.7.19
- Mozilla Firefox Developer Edition versi 63.0b6 (64-bit)

3.3.2 Bahan Penelitian

Bahan yang digunakan selama proses penelitian ini antara lain jurnal, teks buku, *website* serta bahan lainnya yang menunjang selama penelitian.

Daftar Pustaka

Cheng, F. (2011). *Security attack safe mobile and cloud-based one-time password tokens using rubbing encryption algorithm*. *Mobile Networks and Applications* (Vol. 16). https://doi.org/10.1007/s11036-011-0303-9

Cho, S.-I., Lee, H.-J., Lee, S.-G., & Lim, H.-T. (2009). OTP Authentication Protocol Using Stream Cipher with Clock-Counter. *Journal of the Korea Institute of Information and Communication Engineering*, *13*(10), 2113–2120.

Gaw, S., & Felten, E. W. (2006). Password management strategies for online accounts. In *Proceedings of the second symposium on Usable privacy and security* (pp. 44–55).

Huang, Y., Huang, Z., Zhao, H., & Lai, X. (2013). A new One-time Password Method. *IERI Procedia*, *4*, 32–37. https://doi.org/10.1016/j.ieri.2013.11.006

Jindal, P., & Singh, B. (2015). Analyzing the security-performance tradeoff in block ciphers. *International Conference on Computing, Communication and Automation, ICCCA 2015*, 326–331. https://doi.org/10.1109/CCAA.2015.7148425

Konheim, A. G. (2006). *Computer Security and Cryptography*. *Computer Security and Cryptography*. https://doi.org/10.1002/0470083980

Kshemkalyani, A. D., & Singhal, M. (2011). *Distributed computing: principles, algorithms, and systems*. Cambridge University Press.

Kumar, A., & Tiwari, M. N. (2012). effective implementation and avalanche effect of AES. *International Journal of Security, Privacy and Trust Management (IJSPTM)*, *1*(3/4), 31–35.

KumarVerma, H., & Kumar Singh, R. (2012). Performance Analysis of RC6, Blowfish and DES Block Cipher Algorithms. *International Journal of Computer Applications*, *42*(16), 8–14. https://doi.org/10.5120/5774-6004

Lamport, L. (1981). Password Authentification with Insecure Communication. *Communications of the Association for Computing Machinery*, *24*(11), 770–772. https://doi.org/10.1145/358790.358797

Lee, Y. S., Lim, H. T., & Lee, H. J. (2010). A study on efficient {OTP} generation using stream cipher with random digit. *The 12th International Conference on Advanced Communication Technology (ICACT)*, *2*, 1670–1675.

M'raihi, D., Bellare, M., Hoornaert, F., Naccache, D., & Ranen, O. (2005). *Hotp:*

An hmac-based one-time password algorithm.

M'Raihi, D., Machani, S., Pei, M., & Rydell, J. (2011). *Totp: Time-based one-time password algorithm*.

Menezes, A. J., Van Oorschot, P. C., & Vanstone, S. A. (1996). *Handbook of applied cryptography*. CRC press.

Munir, R. (2006). Kriptografi. *Informatika, Bandung*.

Nugroho, E. P., Judhie Putra, R. R., & Ramadhan, I. M. (2017). SMS authentication code generated by Advance Encryption Standard (AES) 256 bits modification algorithm and One time Password (OTP) to activate new applicant account. *Proceeding - 2016 2nd International Conference on Science in Information Technology, ICSITech 2016: Information Science for Green Society and Environment*, 175–180. https://doi.org/10.1109/ICSITech.2016.7852629

Ramanujam, S., & Karuppiah, M. (2011). Designing an algorithm with high Avalanche Effect. *IJCSNS International Journal of Computer Science and Network Security*, *11*(1), 106–111.

Rivest, R. L., Robshaw, M. J. B., Sidney, R., & Yin, Y. L. (1998). The RC6TM block cipher. In *First Advanced Encryption Standard (AES) Conference* (p. 16).

Schneier, B. (2007). *Applied cryptography: protocols, algorithms, and source code in C*. john wiley & sons.

Sommerville, I. (2004). Software Engineering. International computer science series. *Ed: Addison Wesley*.

Sprengers, M. (2011). *GPU-based password cracking: On the security of password hashing schemes regarding advances in graphics processing units*. Master's thesis, Radboud University Nijmegen.

Stallings, W. (2006). *Cryptography and network security: principles and practices*. Pearson Education India.

Sulak, F., Douganaksoy, A., Ege, B., & Koçak, O. (2010). Evaluation of randomness test results for short sequences. In *International Conference on Sequences and Their Applications* (pp. 309–319).

Survey, R. C. A., & Princy, P. (2015). A COMPARISON OF SYMMETRIC KEY ALGORITHMS DES , AES , BLOWFISH , *6*(5), 328–331.

Uymatiao, M. L. T., & Yu, W. E. S. (2014). Time-based OTP authentication via secure tunnel (TOAST): A mobile TOTP scheme using TLS seed exchange and encrypted offline keystore. *ICIST 2014 - Proceedings of 2014 4th IEEE International Conference on Information Science and Technology*, 225–229.

https://doi.org/10.1109/ICIST.2014.6920371

Vinh, T. Le, Bouzefrane, S., Farinone, J. M., Attar, A., & Kennedy, B. P. (2015). Middleware to integrate mobile devices, sensors and cloud computing. *Procedia Computer Science*, *52*(1), 234–243. https://doi.org/10.1016/j.procs.2015.05.061

Wu, M., Garfinkel, S., & Miller, R. (2004). Secure web authentication with mobile phones. In *DIMACS workshop on usable privacy and security software* (Vol. 2010).