BAB II
LANDASAN TEORI

Pada BAB II ini menjelaskan landasan teori yang berkaitan dengan penelitian pengembangan algoritma *OTP* pada pengujian aplikasi pengelola *password* dalam upaya peningkatan kerahasiaan kata sandi.

2.1. *One Time Password*

OTP (One-Time Password) merupakan metode otentikasi menggunakan *password* digunakan sekali saja sehingga setiap kali *login password* yang digunakan akan berubah atau berdasarkan interval waktu yang ditentukan (Huseynov & , Seigneur, 2016) menyebutkan bahwa *OTP* sendiri dapat dibedakan menjadi dua kategori yaitu:

1. *OTP* berbasiskan algoritma matematika

   OTP jenis ini merupakan *OTP* yang menggunakan algoritma matematika kompleks seperti fungsi *hash* kriptografi untuk membangkitkan *password* berdasarkan *password* sebelumnya dan dimulai dari kunci *shared* rahasia.
2. *OTP* berbasis sinkronisasi waktu

   OTP jenis ini *password* berubah berdasarkan sinkronisasi waktu yang berubah secaa konstan pada setiap satuan interval waktu tertentu.

2.2. *HMAC* (*Key-hash Message Authentication code*)

Message Authentication Code merupakan mekanisme membagi knuci rahasia kepada dua pihak agar dapat melakukan autentikasi, ada beberapa jenis *Message Authentication Code*(*MAC*) salah satu contoh adalah *Key-hash Message Authentication code*(*HMAC*) merupakan teknik *MAC* yang menggunakan fungsi *hash* terhadap pesan yang akan dienkrpsi menggnakan kunci privat. *Key-hash Message Authentication code* diciptakan oleh *Mihir Bellare*, *Ran Cannetim*, *Hugo Krawczyk* pada tahun 1996, hal tersebut dikarenakan fungsi *hash* itu sendiri tidak memungkinkan penggunaan kunci untuk menghitung nilai *digest*-nya (Taufik Ramadhany, 2016.).

Pada umumnya algoritma *HMAC* dapat dijelaskan pada persamaan sebagai berikut :

$$HMAC_k\ (m) = h\ ((K\ \oplus opad)\ h\ (K\ \oplus ipad)\ m))) .$$

K merupakan kunci privat yang diketahui oleh pihak pengirim dan pihak penerima, *h* merupakan fungsi *hash* yang digunakan, *m* merupakan pesan yang akan diautentikasi, *opad* (*outer pad*) bilangan konstanta hexadesmial 0x5c5c5c…5c dan *ipad* (*inner pad*) bilangan konstanta hexadesmial 0x363636..36 dengan panjang yang sama.

2.3. *HOTP (HMAC based One Time Password)*

HOTP adalah *HMAC based one time password* merupakan algoritma pembangkit *OTP* yang memanfaatkan fungsi *HMAC*. Standar pada algoritma *HOTP* menggunakan *HMAC* berbasis *SHA*-1 untuk membangkitkan kunci yang digunakan untuk *OTP*. Nilai *OTP* yang dihasilkan merupakan *n*-digit *integer* yang pada fungsi itu dapat diatur. *OTP* menghitung nilai *input* dari *counter* yang nantinya akan di validasi dari pihak *server* dan pihak *client*. Pada *client* akan memiliki kunci rahasia yang berbeda pada masing-masing *client* (Fakhrusy, 2016). Pada penaikan *counter* akan menghasilkan nilai *HOTP* yang berbeda untuk validasi sebagai *user* pada *server*.

Pada *server* dan *client* memiliki *counter* yang saling terintegrasi satu sama lain. Setiap *client* memiliki *counter* yang tidak terintegrasi pada *client* lain sehingga jika seorang *client* pertama sedang mengakses *HOTP*, namun ada *client* lain dengan suatu cara dapat mengakses *HOTP* tersebut maka *client* tersebut bukan mendapatkan *HOTP* melainkan tahapan untuk melakukan autentikasi ini dikarenakan *client* lain tersebut dianggap sebagai *client* yang berbeda oleh *server* maka pada masing-masing *client* memiliki nilai kunci yang unik.

2.4. *TOTP* (*Time-Based One Time Password*)

Time-Based One Time Password (*TOTP*) merupakan algoritma yang dikembangkan dari *HMAC* yang menggabungkan antara *secret key* dengan *current*

timestamp atau waktu saat ini dengan menggunakan fungsi kriptografi hash untuk dapat menghasikan *OTP* sekali pakai. Pada masing-masing *OTP* yang digenerate memiliki batas waktu sehingga dalam kurun waktu tertentu misalnya ketika sistem atau *server* men-*generate password* bisa diakses oleh *user* selama 1 menit maka jika batas waktu telah habis maka *user* tidak dapat mengakses dengan menggunakan *password* sebelumnya namun *server* akan memberikan *password* baru untuk *user*.

Oleh karena itu implementasi *TOTP* sering digunakan pada aplikasi yang membutuhkan validasi *user* kepada *server* misalnya pembayaran *online shop* atau melakukan tahap registrasi pada aplikasi tertentu agar menghindari pencurian data atau penyalahgunaan terhadap akun *user* (Ungkawa et al., 2013). Oleh karena itu dibutuhkan proses autentikasi ketika *user* mengakses layanan dari server, sehingga *server* dapat mengenali *user* dengan *OTP* yang dikirim dari *server* kepada *user*.

2.5. Algoritma *Secure Hash Algorithm* (*SHA*)

SHA merupakan algoritma teknik enkripsi yang dikembangkan pada tahun 1993 oleh *National Insitute and technology (NIST)* dan dipublikasi sebagai *Federal Information Standars* (*FIPS 180*). Pada tahun 1995 ditemukan kelamahan pada algoritma *SHA*-0 sehingga berbagai revisi dan perbaikan dilakukan untuk mengembangkan algoritma yang lebih baik. *SHA*-1 merupakan hasil perbaikan tersebut yang diterbitkan pada tahun 1995. Pada tahun 2001 *SHA*-2 diterbitkan setelah beberapa tahun *SHA*-1 diterbitkan, pada tahun 2006 ditemukan kelemahan pada *SHA*-1 yang rentan terhadap serangan *brute attack* sehingga *NIST* membuat pernyataan untuk mengadopsi penggunaan *SHA*-2 kepada Lembaga federal pada tahun 2010.

NIST mengadakan kompetisi *hash function* untuk membuat sebuah standar *hash* baru yaitu *SHA*-3 pada tahun 2007 dengan melakukan penyerangan *brute force attack*, *brute force attack* merupakan salah satu teknik *cracking password* paling populer yang dilakukan untuk meretas kata sandi, serangan ini dilakukan untuk kemungkinan pesan dapat diketahui pada nilai *hash* (Taufik Ramadhany, 2016.). Meskipun *SHA*-3 dibuat bukan untuk menggantikan *SHA*-2, karena belum ada

Itenas library

serangan hebat yang terjadi pada *SHA*-2 namun atas kekhawatiran *NIST* pada *SHA*-0 dan *SHA*-1 yang berhasil ditembus keamanannya maka *NIST* mencari algoritma *hash* alternatif yaitu *SHA*-3. Algoritma keccak menjadi pemenang dalam kompetisi ini pada tahun 2012, nama keccak sendiri diambil dari nama tarian bali (Kurniawan et al., 2017). Pada hasil keluaran algoritma keccak memiliki keluaran yang beragam. Jenis-jenis keluaran tersebut adalah *SHA*-3 224, *SHA*-3 256, *SHA*-3 384, *SHA*-3 512, *SHAKE*128 dan *SHAKE*256. Algoritma keccak diresmikan sebagai standar baru fungsi *hash* yang dikenal dengan *SHA*-3. Pada komparasi jenis *SHA* ditujukan pada Gambar 2.1 berikut.

Algorithm and variant		Output size (bits)	Internal state size (bits)	Block size (bits)	Rounds	Operations	Security (in bits) against collision attacks	Capacity against length extension attacks	Performance on Skylake (median cpb)[57] long messages	Performance on Skylake (median cpb)[57] 8 bytes	First published
MD5 (as reference)		128	128 (4 × 32)	512	64	And, Xor, Rot, Add (mod 2^{32}), Or	≤18 (collisions found)[58]	0	4.99	55.00	1992
SHA-0		160	160 (5 × 32)	512	80	And, Xor, Rot, Add (mod 2^{32}), Or	<34 (collisions found)	0	≈ SHA-1	≈ SHA-1	1993
SHA-1							<63 (collisions found)[59]		3.47	52.00	1995
SHA-2	*SHA-224*	224	256 (8 × 32)	512	64	And, Xor, Rot, Add (mod 2^{32}), Or, Shr	112	32	7.62	84.50	2004
	SHA-256	256					128	0	7.63	85.25	2001
	SHA-384	384	512 (8 × 64)	1024	80	And, Xor, Rot, Add (mod 2^{64}), Or, Shr	192	128 (≤ 384)	5.12	135.75	2001
	SHA-512	512					256	0	5.06	135.50	
	SHA-512/224	224					112	288	≈ SHA-384	≈ SHA-384	2012
	SHA-512/256	256					128	256			
SHA-3	*SHA3-224*	224	1600 (5 × 5 × 64)	1152	24[90]	And, Xor, Rot, Not	112	448	8.12	154.25	2015
	SHA3-256	256		1088			128	512	8.59	155.50	
	SHA3-384	384		832			192	768	11.06	164.00	
	SHA3-512	512		576			256	1024	15.88	164.00	
	SHAKE128	*d* (arbitrary)		1344			min(*d*/2, 128)	256	7.08	155.25	
	SHAKE256	*d* (arbitrary)		1088			min(*d*/2, 256)	512	8.59	155.50	

Gambar 2.1 Komparasi Jenis *SHA*

(Sumber : https://en.wikipedia.org/wiki/SHA-3/)

2.6. Pengujian *Brute Attack*

Brute attack meruapakan teknik penyerangan terhadap keamanan komputer, serangan *brute attack* terhadap suatu *hash* yang dihasilkan oleh algoritma_*SHA* (Kurniawan et al., 2017). Dengan cara mencoba segala kombinasi huruf, angka dan simbol yang bertujuan untuk mendapatkan *plaintext* dari *hash* tersebut.

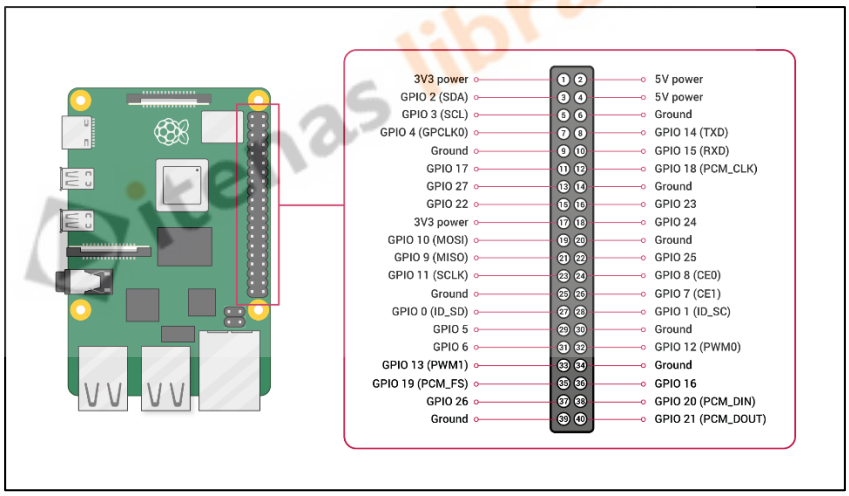
2.7. *IoT* (*Internet of Things*)

IoT (Internet of Things) bisa diartikan seuatu kemampuan untuk membuat segala sesuatu pada mesin misalnya ponsel,mobil dan bahkan kota diharapkan terhubung ke internet memilki perilaku yang cerdas (H., A., & M., 2015)

menyebutkan bahwa dengan mempertimbangkan keberadaan jenis otonomi dan privasi secara umum *IoT* mencakup tiga tuntutan utama yaitu:

1. Pemahaman tetang situasi antara pengguna dan aplikasi.
2. Arsitektur perangkat lunak dan jaringan komunikasi meresap untuk mencakup dan memproses informasi kontekstual.
3. Alat analisis dalam *IoT* yang bertujuan untuk berperilaku otonom dan cerdas.

Pengembangan sering di implementasikan pada mikrokontroller seperti raspberry pi sehingga terciptanya sistem-sistem yang berbasis *IoT*. *Raspberry pi* adalah komputer mini atau sering disebut *SBC(single-board circuit)* yang memiliki ukuran seperti kartu kredit yang berfungsi untuk menjalankan program perkantoran hingga digunakan untuk *server mini*.



Gambar 2.2 *Raspberry pi*

(Sumber : raspberrypi.org/documentation/usage/gpio/)

Perancangan pada perancangan sistem berbasis *IoT* menggunakan *raspberry pi* umumnya dibutuhkan nilai input untuk memberi input maka dibutuhkan sensor, salah satuh contoh sensor yang dapat digunakan oleh *raspberry pi* yaitu sensor ultrasonik yang dihubungkan menggunakan kabel *jumper female to female*, masing-masing *port* akan disesuaikan antara *raspberry pi* dengan sensor untuk *input output*, *ground* dan *volt*.

Gambar 2.3 Sensor Ultrasonik

(Sumber : Ikhsan Maulana., 2017. Mendeteksi Jarak Menggunakan Sensor Ultrasonik HC-SR04 Pada Arduino)

Pada Gambar 2.3 diatas adalah sensor ultrasonik yang dapat memancarkan gelombang untuk mendeteksi jarak pada suatu benda tertentu dengan frekuemsi tertentu. Cara kerja sensor ultrasonik adalah dengan mengubah besaran fisis (bunyi) menjadi besaran listrik dan sebaliknya sehingga menjadi pantulan suatu gelombang dengan kecepatan bunyi 340 m/s.

itenas library