Fingerprint Recognition dengan Knuth-Morris-Pratt *String Matching*

William Stefan Hartono - 13512098
Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung, Jl. Ganesha 10 Bandung 40132, Indonesia
13512098@std.stei.itb.ac.id

***Abstrak* — Konsep *pattern matching* dapat diaplikasikan ke dalam berbagai hal. Salah satu masalah yang berkaitan dengan *pattern matching* adalah untuk mengidentifikasi sebuah *input* yang dimasukkan sebagai suatu pola (*pattern*) dan kemudian mencocokkannya pada basis data.**

Pada *fingerprint recognition*, data hasil pemindaian akan dibandingkan dengan data di dalam basis data digital yang jumlah *byte*-nya berukuran kecil. Sehingga apabila suatu saat dibutuhkan autentikasi, karena ukurannya yang kecil, pencocokan data tidak akan membutuhkan usaha yang besar ataupun lama. Pada makalah ini akan dijelaskan mengenai pencocokan data tersebut dengan menggunakan algoritma pencocokan *string* yaitu algoritma Knuth-Morris-Pratt.

Kata kunci* : autentikasi, sidik jari, *pattern matching

I. PENDAHULUAN

Pada zaman modern ini teknologi semakin berkembang sehingga hampir tidak ada batasan informasi. Hal ini juga menyebabkan semakin berkurangnya keamanan dan privasi. Untuk itu dibuatlah sebuah metode keamanan biometrik dimana hanya ada satu kunci yang benar. Kunci tersebut tidak mudah untuk diduplikasi dan kunci tersebut dapat selalu dibawa-bawa tanpa merepotkan pengguna. Kunci yang digunakan untuk metode keamanan biometrik itu sendiri adalah sebuah biometrik.

Aplikasi biometrik sendiri dapat dikategorikan menjadi tiga buah tujuan umum, yaitu: verifikasi, identifikasi, dan pengecekan duplikasi. Verifikasi secara umum adalah membandingkan secara *one-to-one* terhadap suatu sistem keamanan. Dalam tujuan ini, biometrik biasanya digunakan sebagai sebuah password atau PIN yang kemudian akan dibandingkan dengan data yang ada di basis data. Biasanya verifikasi digunakan untuk penggunaan alat atau aplikasi pribadi, contohnya adalah pada iPhone 5S yang baru dirilis pada bulan September 2013.

Identifikasi secara umum bersifat sama dengan verifikasi. Hanya saja jika pada verifikasi bersifat *one-to-one*, pada identifikasi bersifat *one-to many*. Sistem secara matematis akan melakukan pembandigan dari *input* kepada galeri basis data. Identifikasi biasanya digunakan untuk kepentingan publik, contohnya: investigasi tindak kriminal dan hukum, pembuatan visa, pengecekan latar belakang, dll.

Yang terakhir, pengecekan duplikat bertujuan untuk memeriksa apakah sebuah individu muncul lebih dari satu kali pada basis data. Pengecekan duplikat biasanya diaplikasikan untuk mendeteksi suatu kebohongan/pemalsuan. Contohnya adalah untuk memeriksa apakah suatu individu sudah pernah mengikuti suatu acara pada program social tertentu.

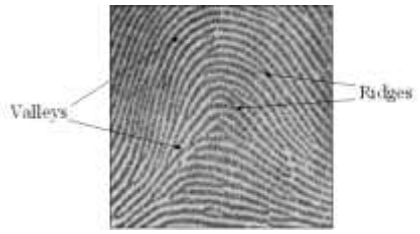
Kunci biometrik adalah kunci yang unik untuk setiap manusia. Sampai saat ini, belum ditemukan dua orang dengan sidik jari yang sama, bahkan pada kasus kembar siam sekalipun. Tidak ada dua buah jari yang identik di dunia ini, bahkan tidak ada dua jari di dunia ini yang ditemukan memiliki kemiripan yang mendekati identik. Sehingga bukti bahwa kunci biometrik adalah kunci yang unik bersifat empiris. Kemudian pada tahun 1999, *Federal Bureau of Intteligence* (FBI) melakukan sebuah percobaan. FBI melakukan sebuah percobaan dimana lima puluh ribu buah sidik jari dibandingkan dengan lima puluh ribu buah sidik jari lainnya. Hasil percobaan tersebut membuktikan bahwa secara matematis tidak mungkin

Kunci biometrik tersebut dapat berupa iris / retina pada bola mata, sidik jari, bau badan, *Deoxyribonucleic acid* (DNA), dan lain-lain. Karena bersifat unik untuk setiap manusia inilah, maka biometrik dapat menjadi satu-satunya kunci yang ada untuk mengakses sesuatu. Hal inilah yang menjadi dasar pengembangan metode keamanan dengan biometrik.

Sesuai dengan namanya, *fingerprint recognition* menggunakan sidik jari sebagai *input* untuk autentikasi. Dasar dari menggunakan sidik jari sebagai sarana autentikasi adalah karena sidik jari unik dan bersifat permanen (walaupun bisa diubah melalui operasi).

Sidik jari manusia terdiri dari dua buah lapisan: lapisan luar (*epidermis*), dan lapisan dalam (*dermis*). Lapisan *dermis* bertanggung jawab untuk menumbuhkan lapisan *epidermis* yang akan menumbuhkan sel-sel ke permukaan jari. Hasilnya adalah sebuah permukaan yang dapat dilihat seperti pada gambar 1. Dengan *ridge* adalah tekstur yang menonjol (puncak) dan *valley* adalah tekstur yang tidak menonjol (lembah). Biasanya *ridge* adalah cerminan dari *valley*, hal ini disebabkan oleh adanya tegangan selama pertumbuhan sel-sel. *Ridge* akan tetap mempertahankan

polanya apabila lapisan *dermis* tidak rusak meskipun terkena goresan. Sehingga meskipun *ridge* tergores/terluka, selama lapisan *dermis* tidak rusak maka luka itu akan tertutup dengan sendirinya. Dan setelah luka itu tertutup, *ridge* akan tumbuh lagi sesuai dengan pola asalnya.

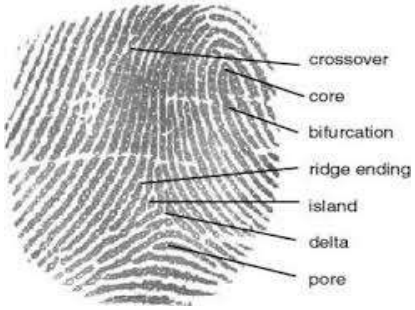


Gambar 1. Permukaan sidik jari
Sumber: http://t2.gstatic.com/

Walaupun sidik jadi bersifat unik untuk setiap manusia, akan tetapi ada pola-pola yang secara umum dapat diidentifikasi. Seorang psikologis dan statistik yang berasal dari Inggris, Sir Francis Dalton mengelompkkan pola-pola sidik jari yang mirip menjadi tiga buah pola dasar. Pola-pola tersebut adalah: *whorl*, *loop*, dan *arch*. Untuk lebih jelasnya, silahkan melihat gambar di bawah.

Gambar 2. Tiga buah pola dasar sidik jari
Sumber:
http://photos1.blogger.com/blogger/3259/1468/1600/*pattern*_types1.jpg

Untuk mengidentifikasi suatu pola sidik jari, dapat dilihat pada bagian *delta* (lihat gambar 3 untuk lebih jelasnya). Bagian *delta* adalah sebuah titik temu dimana *ridge* dari 3 buah arah akan bertemu. Sesuai pada gambar 2, pola *whorl* mempunyai dua buah *delta* (di sebelah kiri bawah dan di sebelah kanan bawah). Pola *loop* mempunyai satu buah *delta* (di sebelah kiri bawah) dan pola *arch* tidak mempunyai bagian *delta* sama sekali. Selain ketiga pola diatas, masih banyak terdapat pola-pola sidik jari yang lain, contohnya: *double loop*, *ulnar loop*, *radial loop*, *peacock's eye*, *tented arch*, *accidental*, dan lain-lain.



Gambar 3. Penjelasan bagian sidik jari
Sumber:
http://www.griaulebiometriks.com/images/books/understanding/*fingerprint*formation.bmp

Pada umumnya pola sidik jari dapat langsung dibedakan jika kita melihat bagian tengah jari dan bagian *delta*. Akan tetapi, tidak demikian halnya dengan bagian tepi jari. Pola pada bagian tepi jari susah dibedakan satu sama lain karena hampir semua sidik jari mempunyai aliran *ridge* yang sama di batas sidik jari.

II. *FINGERPRINT RECOGNITION*

Alat dan sensor yang digunakan untuk melakukan *biometric recognition* biasanya alat mekanik atau sistem elektronik yang dapat mendigitalisasi dan mengonversikan *input* ke dalam bentuk *template* biometrik. Dalam kasus untuk *fingerprint recognition*, kebanyakan sensor biasanya berupa sensor optik atau sensor yang bersifat kapasitif.

Sensor kapasitif dapat berupa *scanner* yang men-*scan* satu ujung jari penuh atau dapat men-*scan* jari yang digeser (*swipe*). Sensor ini biasanya terbuat dari sebuah *chip* berbahan dasar silikon yang dapat mendeteksi arus listrik pada *ridge* sidik jari ketika ada sebuah kontak. Sedangkan untuk sensor *optik*, digunakan kombinasi sumber cahaya, sebuah prisma, dan sebuah sensor cahaya untuk mendapatkan gambar sidik jari. Yang perlu diperhatikan dari alat pendeteksi dan sensor yang digunakan adalah hasilnya harus mempunyai resolusi yang cukup, kontras, bebas dari distorsi, dan dapat dikompres sesuai kebutuhan basis data.

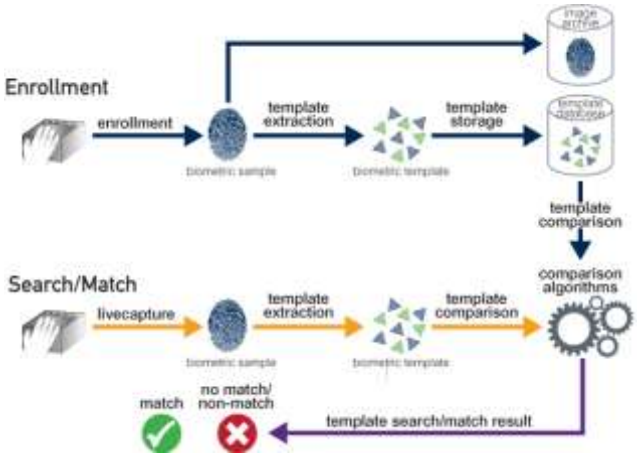
Sistem sidik jari sendiri biasanya melibatkan proses: pendaftaran, *live capture*, ekstrasi *template*, dan perbandingan *template*. Tujuan dari pendaftaran adalah untuk mengumpulkan dan menyimpan data sidik jari serta untuk membuat *template* kombinasi angka untuk perbandingan di masa depan. Dengan mengarsipkan data baru/mentah, *template* yang baru dapat dibuat jika pada suatu saat digunakan algoritma pencocokan yang baru.

Live capture bertujuan untuk mengumpulkan sampel biometrik pada saat akses atau identifikasi dan membandingkannya dengan galeri data yang sudah tersimpan pada langkah pendaftaran.

Ekstrasi *template* bertujuan untuk menciptakan *template* kombinasi angka dari data mentah yang didapat pada saat *live capture* dan enrollment. Biasanya hal ini dilakukan pada saat enrollment untuk menghemat waktu.

Perbandingan *template* bertujuan untuk mendapatkan suatu nilai kecocokan. Perbandingan data ini biasanya menggunakan suatu algoritma *pattern*/*string matching*. Pada saat perbandingan selesai dilakukan, sebuah nilai kecocokan akan dihasilkan. Nilai kecocokan tersebut kemudian ditentukan dengan nilai batas yang sudah ditentukan. Jika nilai kecocokan lebih besar sama dengan nilai batas, maka sidik jari tersebut benar/sudah terdaftar dan sebaliknya.

Berikut penulis tampilkan gambar untuk memudahkan pembaca mengerti kronologis sistem sidik jari.



Gambar 4. Kronologis sistem autentikasi sidik jari
Sumber: http://www.aware.com/biometriks/whitepapers/wab_biometrik-processes.html

III. ALGORITMA KNUTH-MORRIS-PRATT (KMP)

Ada banyak algoritma *pattern*/*string matching*, seperti algoritma brute force, algoritma Boyer-Moore (BM), algoritma Knuth-Morris-Pratt (KMP), dan lain-lain. Akan tetapi, makalah ini akan membahas *string matching* dengan algoritma Knuth-Morris-Pratt.

Sesuai dengan namanya, algoritma ini ditemukan olah Donald Knuth, Vaughan Pratt, dan James H. Morris. Masing-masing dari nama di atas menemukan algoritma ini pada tahun 1974. Kemudian, pada tahun 1977 ketiganya bergabung dan mempublikasikannya. Oleh karena itu algoritma ini bernama Knuth-Morris-Pratt.

Algoritmanya adalah sebagai berikut:

Diberikan T (*Text*) bertipe *string* dengan panjang n karakter dan P (*Pattern*) bertipe *string* dengan panjang m karakter. Asumsi m kurang dari sama dengan n. Algoritma ini akan mencari P dalam T dari kiri ke kanan seperti algoritma *brute force* pada umumnya. Akan tetapi, perbedaan algoritma ini dengan algoritma *brute force* adalah algoritma ini akan berpindah/menggeser P dengan cara yang lebih cerdik dibandingkan dengan algoritma *brute force*.

Jika terjadi sebuah ketidakcocokan pada T dan P, maka akan dicari *prefix* terbesar dari P yang sekaligus merupakan *suffix* dari P. Jika ditemukan, maka P akan digeser sebanyak panjang P sendiri dikurangi dengan panjang *prefix* terbesar dari P yang sekaligus merupakan *suffix* dari P.

Perhitungan jumlah pergeseran P dilakukan dengan algoritma KMP *Border Function* atau yang dikenal juga dengn nama *Failure Function*. Algoritma ini akan *memroses* P terlebih dahulu untuk menghitung berapa panjang *prefix* terbesar dari P yang sekaligus merupakan *suffix* dari P.

Kompleksitas waktu KMP berada pada orde O(m + n). Hal ini didapat dari orde O(m) untuk menghitung KMP *Border Function*, dan O(n) untuk mencari *string*. Karena ordenya O(m + n), algoritma KMP jauh lebih cepat dibandingkan dengan algoritma *brute force* yang memiliki kompleksitas O(mn).

Algoritma KMP memiliki keuntungan yaitu bahwa algoritma ini tidak akan pernah melakukan pencarian mundur (*backwards*) pada T. Hal ini mengakibatkan algoritma ini cocok untuk memroses *file* yang berukuran besar. Akan tetapi, algoritma ini memiliki kelemahan yaitu algoritma ini tidak akan berjalan dengan optimal saat P panjang. Hal ini disebabkan karena meningkatnya kemungkinan terjadinya ketidakcocokan. Terlebih lagi apabila ketidakcocokan tersebut terjadi di awal P (jika terjadi di akhir P, P akan lebih banyak digeser sehingga mempercepat pencarian).

Berikut penulis tampilkan kode KMP dan algoritma Border Function dalam bahasa pemrograman Java:

Algoritma KMP

```
publik static int KMP(String text, String
pattern)
        {
                // membuat pencarian menjadi case
insensitive
                String inputtext = text;
inputtext = inputtext.toUpperCase();
                String inputpattern = pattern;
inputpattern = inputpattern.toUpperCase();

                int n = inputtext.length();     //
n adalah panjang text
                int m = inputpattern.length(); //
m adalah panjang pola
                int fail[] =
computeFail(pattern);

                int i=0;
                int j=0;

                while(i < n)
                {
                        if(inputpattern.charAt(j)
== inputtext.charAt(i))
                        {
                                if (j == m - 1)
                                {
                                        return i -
m + 1;
                                }
                                i++;
                                j++;
                        }
                        else if (j > 0)
                        {
                                j = fail[j-1];
                        }
                        else
                        {
                                i++;
                        }
                }
                return -1; // tidak ada kecocokan
        }
```

Sumber: Dr. Andrew Davison WiG Lab (teachers room), CoE ad@fivedots.coe.psu.ac.th; 240-301 Computer Engineering Lab III(Software) Semester 1, 2006-2007

Algoritma *Border Function*

```
publik static int[] computeFail(String pattern)
        {
                int fail[] = new
int[pattern.length()];
                fail[0] = 0;
```

```
                int m = pattern.length();
                int j = 0;
                int i = 1;

                while(i < m)
                {
                        if  (pattern.charAt(j) ==
pattern.charAt(i)) // ditemukan kecocokan pada j
+ 1
                        {
                                fail[i] = j + 1;
                                i++;
                                j++;
                        }
                        else if (j > 0) // j cocok
dengan prefix
                        {
                                j = fail[j - 1];
                        }
                        else // tidak ada
kecocokan
                        {
                                fail[i] = 0;
                                i++;
                        }
                }
                return fail;
}
```

Untuk lebih jelasnya, silahkan perhatikan contoh berikut:

Gambar 5. Contoh penerapan algoritma KMP
Sumber: Dr. Andrew Davison WiG Lab (teachers room), CoE ad@fivedots.coe.psu.ac.th; 240-301 Computer Engineering Lab III(Software) Semester 1, 2006-2007

Pertama-tama, posisi karakter pertama T dan P akan disamakan. Kemudian pengecekan akan dilakukan. Karena karakter tersebut cocok, maka pengecekan akan dilanjutkan ke karakter berikutnya. Hal ini akan terus berulang hingga pada posisi ke-5. Pada posisi ke-6, karakter pada T dan P tidak cocok.

Pada saat terjadi ketidakcocokan ini, kita mengetahui bahwa 5 karakter pertama pada P sudah cocok dengan 5 karakter pertama pada T. Karakter P yang sudah cocok hingga saat ini adalah "abaca". Sehingga dari "abaca" tersebut didapat himpunan *prefix* dari P yaitu: {"a", "ab", "aba", "abac"}. Sedangkan himpunan *suffix* dari P yaitu: {"a", "ca", "aca", "baca"}. Dari himpunan *prefix* dan himpunan *suffix* yang didapat, kita dapat mencari *prefix* terpanjang yang sekaligus merupakan *suffix* dari P atau sebaliknya. Dengan melihat himpunan yang sudah didapat, dapat langsung diketahui bahwa yang memenuhi adalah "a" yang mempunyai panjang karakter = 1.

Setelah didapat *prefix* terpanjang yang sekaligus merupakan *suffix* dari P, kita dapat menghitung seberapa jauh kita harus menggeser P.

Jumlah pergeseran P = jumlah karakter P yang sudah dicocokkan dan benar – jumlah karakter *prefix* terpanjang yang sekaligus merupakan *suffix* dari P.

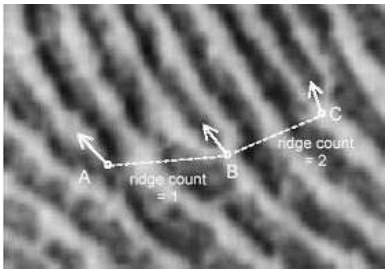
Dalam kasus pada contoh berarti P digeser sebanyak = 5 – 1 = 4. Oleh karena itu, karakter pertama P sekarang berada pada posisi ke-5. Hal ini terus diulang hingga pada akhirnya tidak tersisa lagi karakter pada T untuk dicocokkan dengan P (tidak ditemukan hasil) atau ditemukan sebuah kecocokan. Jika ditemukan sebuah kecocokan, maka algoritma yang ditampilkan akan mengembalikan posisi dimana karakter pertama ditemukan.

Pada gambar 5 terdapat tabel di pojok kiri bawah. Pada tabel itu k adalah jumlah karakter P. Sedangkan b(k) berarti: jumlah karakter *prefix* terpanjang yang sekaligus merupakan *suffix* dari P, dengan batasan P[1..k].

IV. IMPLEMENTASI PADA PROGRAM

Pada saat pencocokan dilakukan, sebelumnya akan ditentukan terlebih dahulu bentuk umum dari *live capture* yang didapat. Seperti yang sudah dijelaskan pada bab sebelumnya, bentuk umum mencakup: *whorl*, *loop*, dan *arch*. Saat bentuk umum dari *live capture* sudah ditentukan, hal ini akan mempermudah pencarian karena kita hanya perlu mencari di dalam basis data sidik jari yang memiliki bentuk umum yang sama dengan *live capture*.

Beberapa algoritma menggunakan pola-pola/sifat-sifat sidik jari yang spesifik seperti contohnya pada Gambar 2 yaitu: *bifurcation*, *ridge ending*, dan lain-lain. Beberapa algoritma yang lain menghitung jumlah *ridge* antara 2 titik yang sudah ditentukan. Untuk lebih jelasnya, silahkan lihat Gambar 6.

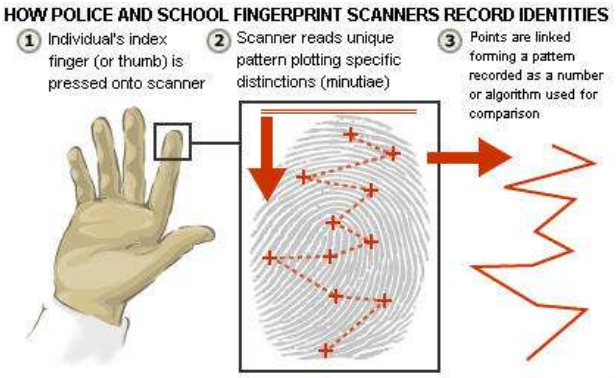


Gambar 6. Penghitungan *ridge* antara titik yang sudah ditentukan

Pada gambar di atas, jumlah *ridge* di antara titik A dan titik B adalah satu, sedangkan jumlah *ridge* di antara titik B dan titik C adalah dua.

Pada algoritma *pattern/string matching*, yang digunakan adalah bentuk umum dari *ridge*. Hasil dari *live capture* yang didapat akan dibagi menjadi beberapa sektor, dimana informasi-informasi seperti arah mengaliranya *ridge* dan lain-lain akan diekstrak dan disimpan pada basis data. Untuk lebih jelasnya, silahkan

melihat Gambar 7.



Gambar 7. Penyimpanan pola sidik jari sebagai template.
Sumber: http://www.identityblog.com/wp-content/images/2007/03/fingerprint-template.jpg

Pada saat pencocokan ingin dilakukan, *scanner* akan menggunakan (*one-way function*) untuk mengubah hasil *live capture* sehingga cocok dengan *template* dengan batas toleransi (disebabkan adanya faktor alat / faktor manusia dan lain-lain yang mungkin menyebabkan ketidakakuratan hasil *scan*).

Akan tetapi, karena adanya toleransi, di beberapa sistem, t*emplate* yang dihasilkan mungkin tidak bertindak sebagai *key* yang bisa dengan mudah langsung dicocokkan (exact match). Oleh sebab itu, sistem akan mencari sejumlah *template* yang kira-kira cocok dari basis data dan melakukan kalkulasi pencocokan.

Kemudian, setelah menyeleksi dari basis data sidik jari mana saja yang harus dijadikan sebagai bahan perbandingan, algoritma KMP akan dijalankan sebagai dasar cara untuk membandingkan *live capture* dengan kumpulan sidik jari pada basis data.

V. PENGUJIAN KEAKURATAN SISTEM

Sistem bisa saja tidak berjalan sesuai dengan yang diinginkan oleh pengguna. Hal ini dapat terjadi karena adanya berbagai sumber *error*, seperti: faktor manusia, faktor alat, dan faktor algoritma. Faktor manusia mencakup perilaku manusia pada saat pengambilan *live capture* dilakukan. Faktor alat mencakup tingkat responsivitas dan sensitivitas serta komponen-komponen dari alat tersebut. Faktor algoritma mencakup algoritma yang digunakan, komponen/langkah pendaftaran, komponen/langkah perbandingan.

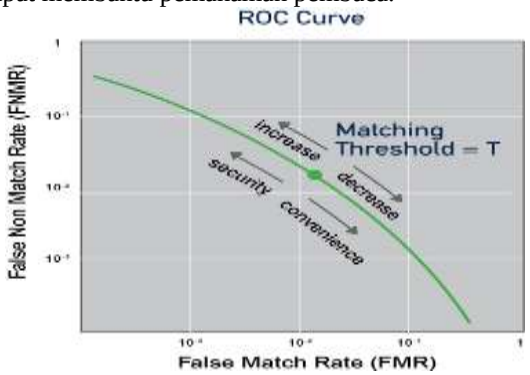
Dengan sistem seperti yang sudah dijabarkan pada bab sebelumnya, kita dapat melakukan pengujian terhadap keakuratan sistem sidik jari. Dari hasil percobaan diharapkan didapatkan nilai rata-rata keakuratan sistem. Untuk melakukan pengujian terhadap sistem, kita harus menentukan berapa banyak jumlah percobaan yang harus dilakukan terlebih dahulu. Jumlah percobaan harus ditentukan sejak awal karena pengujian membutuhkan/menghabiskan banyak uang, waktu dan sumber daya. Oleh sebab itu jumlah pengujian yang dilakukan terhadap sistem tidak perlu terlalu banyak. Akan tetapi, jumlah pengujian juga tidak dapat terlalu sedikit karena pengujian tersebut harus bersifat kaku/tidak mudah berubah hasilnya dan dapat mewakili nilai rata-rata keberhasilan sistem yang sesungguhnya.

Setelah menentukan jumlah pengujian yang akan dilakukan, ada beberapa parameter yang dapat digunakan untuk menguji sistem pencocokan sidik jari. Beberapa parameter tersebut contohnya adalah: tingkat penerimaan pengguna terhadap sistem, tingkat pengulangan sistem, waktu respon sistem, dan lain-lain.

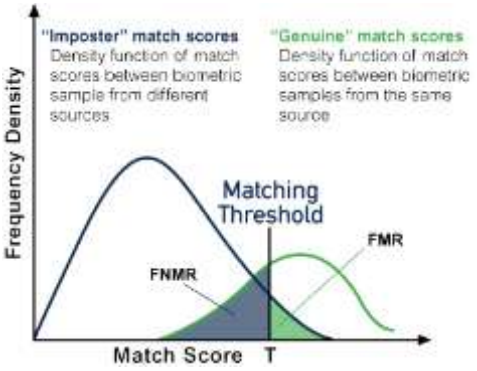
Tingkat keakuratan sistem biometrik sendiri ditentukan oleh kurva *receiver operating characteristic* (ROC) yang memetakan *false match rate* (FMR) dan *false non-match rate* (FNMR) terhadap sampel-sampel yang terdapat di dalam basis data. FMR adalah jumlah kesalahan dimana sampel dari sumber yang berbeda dinilai secara salah dari sumber yang sama (ketika sesuatu yang benar dianggap salah). Sedangkan FNMR adalah jumlah kesalahan dimana sampel dari sumber yang sama dinilai secara salah dari sumber yang berbeda (kebalikan dari FMR, dimana sesuatu yang salah dianggap benar).

Sistem sangat bergantung terhadap basis data yang ada, oleh sebab itu, dengan sampel yang sama dapat dihasilkan ROC yang berbeda-beda. Beberapa algoritma dapat "dilatih" untuk bekerja dengan lebih efisien. Hal ini dapat dibandingkan dengan contoh saat mahasiswa melihat soal ujian sebelum berusaha menjawabnya. Mahasiswa tersebut dapat memilih terlebih dahulu soal mana yang akan dikerjakan. Algoritma yang sudah "dilatih" tentunya akan bekerja dengan lebih efisien, tetapi hal ini tidak merubah performansi sistem pada data-data yang asing. Sehingga langkah terbaik untuk melakukan pengujian keakuratan pada sistem adalah dengan mencobai sistem dengan data-data yang asing bagi sistem, dimana algoritma yang sudah "dilatih" tersebut tidak akan berfungsi dengan optimal.

Berikut penulis menampilkan gambar yang diharapkan dapat membantu pemahaman pembaca.



Gambar 8. Kurva ROC
Sumber:
http://www.aware.com/biometriks/whitepapers/wab_biometrik-system-accuracy-testing.html



Gambar 9. Fungsi densitas yang menggambarkan perbandingan FMR dan FNMR
Sumber: http://www.aware.com/biometriks/whitepapers/wab_biometrik-system-accuracy-testing.html

VI. KEUNTUNGAN DAN KEKURANGAN SISTEM

Dalam penerapan algoritma KMP untuk *fingerprint recognition*, terdapat beberapa keuntungan dan kerugian.

Beberapa keuntungannya antara lain:

- *template* yang digunakan untuk menyimpan data sidik jari relatif kecil (walaupun lebih besar daripada retinal *recognition*),
- *template* yang relatif kecil menyebabkan pencocokan yang relatif cepat,
- yang dijadikan sampel biometrik adalah sidik jari yang dapat digantikan kapan pun dengan mudah (bila terjadi kerusakan pada sidik jari, sidik jari dapat beregenerasi dengan cukup cepat, selain itu basis data untuk setiap individu tidak terbatas hanya pada satu jari, dapat lebih),
- tidak terdapat resiko terpengaruh/rusaknya sidik jari oleh alat maupun cara pemindaian yang digunakan,
- penggunaan teknologi *fingerprint recognition* pada saat ini sudah mulai semarak,
- karena berupa sistem digital, kita dapat mengetahui data dan statistik usaha autentikasi, persentasi keberhasilan, dan lain-lain,

Meskipun memiliki cukup banyak keuntunga, *fingerprint recognition* dengan algoritma KMP juga memiliki beberapa kerugian, beberapa di antaranya adalah

- harga teknologi dan biaya perawatan yang relatif masih mahal,
- dibutuhkan tenaga ahli untuk melakukan setting pada sistem secara berkala, sekadar orang awam tidak disarankan,
- kebergantungn sistem yang besar pada terhadap listrik, faktor manusia, faktor alat, basis data, dan faktor algoritma,
- adanya kemiripan data (pola sidik jari) yang banyak dapat mengurangi keoptimalan kerja algoritma KMP,
- sistem mengubah sidik jari dari *live capture* sebagai *template* untuk disimpan pada basis data yang menyebabkan kita dapat menciptakan sebuah *template*, tetapi tidak dapat mengubah lagi dari *template* sebagai sidik jari (*one-way function*)

VII. KESIMPULAN

Algoritma KMP dapat diterapkan pada banyak hal, salah satunya adalah sebagai cara untuk mengidentifikasi/melakukan proses autentikasi pada sistem keamanan *fingerprint recognition*. Performansi dari KMP sendiri sangat bergantung dari *live capture* yang didapat sekaligus pada basis data.

Sidik jari merupakan salah satu bagian tubuh manusia yang relatif cepat dalam hal regenerasi dan dapat digantikan dengan sidik jari pada jari lainnya. Selain itu penggunaan sidik jari sebagai sarana autentikasi sendiri tidak memiliki resiko terhadap kesehatan pengguna sehingga *fingerprint recognition* relatif mudah dan aman untuk diterapkan.

Fingerprint recognition memanfaatkan pola yang unik pada setiap individu yang dibentuk oleh susunan-susunan dan aliran-aliran *ridge* pada ujung jari. Terdapat tiga pola sidik jari yang umum, yaitu: *whorl*, *loop*, dan *arch*.

Sistem sidik jari sendiri biasanya melibatkan proses: pendaftaran, *live capture*, ekstrasi *template*, dan perbandingan *template*. Pencocokan pola sidik jari yang didapat pada *live capture* dapat dilakukan dengan membandingkan *template* pada basis data. Sebelum iu, hasil dari *live capture* harus dikonversi terlebih dahulu menjadi kombinasi angka sehingga sesuai dengan *template*.

Sistem mungkin saja tidak memberikan hasil yang sesuai dengan harapan pengguna. Hasil ini dapat diteliti lebih lanjut dengan menggunakan kurva ROC, FMR, dan FNMR.

VIII. SARAN TERHADAP PENGEMBANGAN SISTEM

Sistem ini dapat dibilang sangat sederhana dan memiliki beberapa kekurangan. Akan tetapi, sistem dapat diperbaiki dengan:

- menambahkan algoritma yang lain seperti Boyer-Moore (BM), algoritma berbasis gambar, algoritma berbasis ekstrasi minutia, dan lain-lain
- menurunkan nilai ambang batas kecocokan untuk verifikasi jika dirasa cukup aman (sistem melakukan banyak kesalahan dalam melakukan verifikasi), dapat juga dilakukan dengan menerapkan *approximation match* sebagai ganti dari *exact match*

IX. UCAPAN TERIMA KASIH

Penulis mengucapkan terima kasih kepada Tuhan Yang Maha Esa, karena atas rahmat-Nya penulis dapat menyelesaikan makalah ini sebagaimana mestinya. Penulis juga mengucapkan terima kasih kepada Bapak Rinaldi Munir, dan Bu Masayu sebagai dosen pengajar mata

kuliah IF2211 Strategi Algoritma yang telah memberikan banyak pelajaran selama ini. Tidak lupa penulis juga mengucapkan terima kasih kepada semua pihak yang telah memberikan saran dan koreksi, yang tidak dapat disebutkan satu – persatu.

X. DAFTAR REFERENSI

Fikri, Abdurrisyad. 2010. *Pencocokan Pola Retina (Retinal Recognition) dengan Algoritma Pencocokan String Knuth-Morris-Pratt*

Fisitania, Azalea. 2013. *Pattern Matching for Detecting Plagiarism on Artworks*

http://galton.org/ diakses pada 10 Mei 2014 pukul 18:00

http://www.aware.com/biometriks/whitepapers/wab_biometrik-applications.html diakses pada 10 Mei 2014 pukul 07:35

http://www.aware.com/biometriks/whitepapers/wab_biometrik-processes.html diakses pada 10 Mei 2014 pukul 07:34

http://www.aware.com/biometriks/whitepapers/wab_biometrik-system-accuracy-testing.html diakses pada 10 Mei 2014 pukul 07:34

http://www.aware.com/biometriks/whitepapers/wab_devices-sensors.html diakses pada 10 Mei 2014, 07:36

http://www.bioconsulting.com/How_Accurate_Is_The_Biometrik.htm diakses pada 14 Mei 2014 pukul 9:23

http://www.cse.msu.edu/~rossarun/pubs/FengJainRoss_Altered*Fingerprint*_TechReport09.pdf diakses pada 10 Mei 2014 pukul 07:37

http://www.griaulebiometriks.com/en-us/book/understanding-biometriks/evaluation/accuracy/error diakses pada 14 Mei 2014 pukul 9:48

http://www.latent-prints.com/Thornton.htm diakses pada 10 Mei 2014 pukul 07:20

Munir, Rinaldi. 2009. Diktat Kuliah IF2211 Strategi Algoritma. Bandung: Program Studi Teknik Informatika

XI. PERNYATAAN

Dengan ini saya menyatakan bahwa makalah yang saya tulis ini adalah tulisan saya sendiri, bukan saduran, atau terjemahan dari makalah orang lain, dan bukan plagiasi.

Bandung, 13 Mei 2014

William Stefan Hartono
13512098