Pengaplikasian Algoritma *Depth-First Search* dan Pencocokan *String* untuk Memverifikasi Keabsahan File Nandroid *Backup* dalam Sistem *Recovery* di Android

Asep Saepudin – 13511093[1]
Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung, Jl. Ganesha 10 Bandung 40132, Indonesia
[1]asepsaepudin@s.itb.ac.id

Saat ini, lebih dari delapan puluh persen sistem operasi pada ponsel pintar didominasi oleh Android. Sifat Android yang open source membuat sistem operasi ini bisa dengan mudah ditemui dari ponsel pintar yang low-end hingga high-end. Tidak hanya ponsel pintar, beberapa perangkat lain seperti mesin cuci, konsol permainan, jam tangan, bahkan kacamata pun sudah mulai mengadopsi Android. Salah satu kelebihan yang dimiliki Android adalah sifatnya yang mudah dimodifikasi. Bagi pengguna Android yang sering memodifikasi sistem operasinya, tentunya tidak asing mendengar kata Nandroid Backup and Restore. Nandroid Backup and Restore adalah suatu prosedur yang akan mem-backup atau me-restore data yang ada di file sistem Android. File-file yang berada di file sistem Android memiliki kendali yang lebih terhadap data-data yang ada di Android karena memiliki privilege khusus. Oleh karena itu, dalam melakukan backup dan restore tentunya kita harus mengecek integritas file backup yang telah dibuat untuk memastikan bahwa file backup tersebut tidak mengandung file yang tidak diinginkan yang dapat membawa sistem Android ke kondisi yang berbahaya, misal malware, virus, spyware, dll. Dalam makalah ini, akan dipaparkan pengaplikasian algoritma DFS dan pencocokan string untuk memverifikasi nandroid backup and restore dalam sistem recovery di Android.

Kata Kunci: algoritma DFS, Android, nandroid backup and restore, pencocokan string.

I. PENDAHULUAN

Android adalah suatu sistem operasi yang berbasis Linux. Android, sama seperti istilah *superuser* pada Linux, memiliki istilah *root* yang mendeskripsikan bahwa suatu proses dijalankan dengan *privilege* tertinggi. Itu artinya, proses tersebut memiliki akses untuk membaca, mengedit, menghapus file dan folder yang ada di file sistem Android. Hal itulah yang menjadi kelebihan sekaligus kekurangan Android. Disebut suatu kelebihan karena dengan hal tersebut, pengguna bisa memodifikasi file sistem untuk membuat kinerja Android lebih optimal, mengedit *user interface* seperti yang kita inginkan, menambahkan fitur yang secara *default* belum ada, dll. Disebut suatu kekurangan karena dengan hal tersebut, apabila ada suatu proses yang tidak diinginkan berjalan di kondisi *root*, maka proses tersebut bisa mencuri informasi penting yang ada file sistem, menghapus file penting dari file sistem, dll.

Namun kekurangan tersebut nyatanya tidak menghalangi banyak orang untuk mencoba memodifikasi Android. Dalam memodifikasi sistem operasi Android, tidak jarang orang-orang melakukan kesalahan sehingga ketika sistem operasi yang telah dimodifikasi tersebut dicoba untuk diimplementasikan di devais berbasis Android, devais tersebut tidak mau menyala atau mampu menyala namun dalam kondisi *stuck/loop* akibat konfigurasi file sistem yang tidak tepat. Oleh karena itu sangat penting bagi tiap orang yang ingin memodifikasi Androidnya untuk melakukan proses *backup* terhadap file sistemnya. *Backup* ini berguna apabila pengguna mengalami kondisi di atas. Hanya dengan melakukan proses *restore*, maka ponsel akan kembali ke kondisi saat ponsel di-*backup*, sama persis. Cara paling umum untuk melakukan backup file sistem adalah dengan menggunakan fitur nandroid *backup* yang ada di sistem *custom recovery* Android.

Custom recovery adalah suatu sistem *recovery* yang ada di Android yang dibuat oleh pihak ketiga. Sistem *recovery* ini memiliki kelebihan dibanding sistem *recovery* yang terinstal secara *default* oleh vendor.[1] Kelebihan-kelebihan tersebut antara lain adalah:

- Membuat *backup images*. Fungsi ini kan mem-*backup* ini semua file sistem, pengaturan ponsel, aplikasi yang terinstal, data sms, nomor kontak, bahkan kernel dan setiap aspek dari keaadaan devais saat melakukan backup. Proses *backup* dengan menggunakan custom recovery disebut nandroid *backup* karena *backup* menggunakan *custom recovery* mampu menjangkau partisi NAND *flash* pada devais Android.
- Me-*restore* backup images. Fungsi ini adalah untuk mengembalikan keadaan ponsel sama seperti keadaan devais saat *backup* dilakukan.
- *Factory data reset*. Fungsi ini akan menghapus seluruh pengaturan yang ada sehingga keadaan ponsel akan kembali seperti dalam keadaan baru.
- Partisi SD Card. Fungsi ini memungkinkan pengguna untuk mempartisi sd card menjadi beberapa bagian,

format file sistem yang didukung mulai dari ext2, ext3, dan ext4.

- Dll.

Gambar 1 ClockWorkMod Custom Recovery (http://techxtras.blogspot.com/2013/04/clockworkmod-cwm-recovery-and-twrp.html)

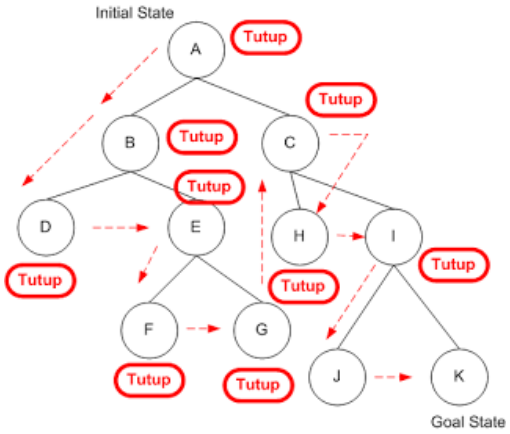
Sama seperti sistem operasi lainnya, Android memiliki struktur file sistem yang dapat digambarkan dalam suatu graf berbentuk pohon. File sistem inilah yang menjamin keberlangsungan berjalannya fungsi-fungsi pada sistem operasi Android. Ketika proses nandroid *backup* berlangsung, *custom recovery* akan menyalin *state* file sistem sama persis. Permasalah terjadi ketika ada pihak yang tidak berwenang mencoba untuk mengubah file hasil *backup* dengan menambahkan file-file yang tidak diinginkan seperti virus, *malware*, *adware*, dll. Oleh karena itu diperlukan adanya pengecekan integritas file hasil *backup* untuk mencegah hal ini terjadi. Salah satu caranya adalah dengan menggunakan algoritma DFS untuk menandai *state* file yang di-*backup* dan pencocokan string untuk membandingkan kesamaan file hasil *backup* dengan *state* yang telah dibuat.

II. DASAR TEORI

A. Algoritma DFS

Algoritma *Depth-First Search* (DFS) adalah salah satu algoritma pencarian solusi yang digunakan di dalam kecerdasan buatan. Algoritma ini termasuk salah satu jenis algoritma yang melakukan pencarian dalam urutan tertentu tetapi tidak memiliki informasi apa-apa sebagai dasar pencarian kecuali hanya mengikuti pola yang diberikan.[2]

Di dalam DFS, pencarian dilakukan pada suatu struktur pohon yaitu kumpulan semua kondisi yang mungkin yang diimplementasikan dalam sebuah struktur pohon. Paling atas adalah akar (*root*) yang berisi kondisi awal pencarian (*initial state*) dan di bawahnya adalah kondisi-kondisi berikutnya sampai kepada kondisi tujuan (*goal state*).



Gambar 2 Ilustrasi Graf (http://cikalinspirasi.blogspot.com/2013/05/algoritma-dept-first-search-dfs.html)

Untuk melakukan pencarian, DFS menggunakan cara sebagai berikut :

1. Kunjungi simpul v,
2. Kunjungi simpul w yang bertetangga dengan simpul v.
3. Ulangi DFS mulai dari simpul w.
4. Ketika mencapai simpul u sedemikian sehingga semua simpul yang bertetangga dengannya telah dikunjungi, pencarian dirunut-balik (*backtrack*) ke simpul terakhir yang dikunjungi sebelumnya dan mempunyai simpul w yang belum dikunjungi.
5. Pencarian berakhir bila tidak ada lagi simpul yang belum dikunjungi yang dapat dicapai dari simpul yang telah dikunjungi.[3]

B. Pencocokan *String*

Pengertian *string* menurut Dictionary of Algorithms and Data Structures, National Institute of Standards and Technology (NIST) adalah susunan dari karakter-karakter (angka, alfabet atau karakter yang lain) dan biasanya direpresentasikan sebagai struktur data larik. *String* dapat berupa kata, frase, atau kalimat.

Pencocokan string merupakan bagian penting dari sebuah proses pencarian string (*string searching*) dalam sebuah dokumen. Hasil dari pencarian sebuah *string* dalam dokumen tergantung dari teknik atau cara pencocokan *string* yang digunakan. Pencocokan *string* (*string matching*) menurut Dictionary of Algorithms and Data Structures, National Institute of Standards and Technology (NIST), diartikan sebagai sebuah permasalahan untuk menemukan pola susunan karakter *string* di dalam *string* lain atau bagian dari isi teks.[4]

Dalam pencocokan string dikenal istilah *teks* dan *pattern*. Teks atau text yaitu *string* yang panjangnya sebanyak n karakter sedangkan *pattern* yaitu *string* dengan panjang m karakter dimana m <= n yang akan dicari di dalam teks.

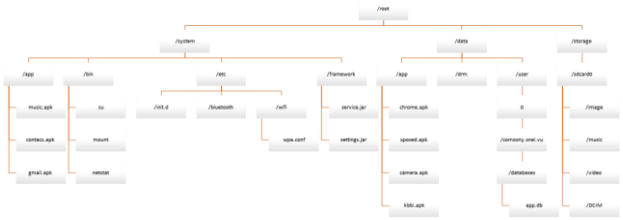
Contoh -Teks : Di luar sedang hujan deras.
-Pattern : hujan

Salah satu cabang dari klasifikasi pencocokan string adalah *exact string matching* yang merupakan pencocokan *string* secara tepat dengan susunan karakter dalam *string* yang dicocokkan memiliki jumlah maupun urutan karakter dalam *string* yang sama.

Contoh : kata *android* akan menunjukkan kecocokan hanya dengan kata *android*.

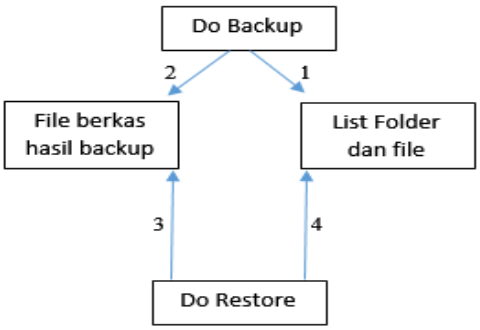
III. IMPLEMENTASI

Seperti yang telah disinggung sebelumnya, Android memiliki file sistem yang dapat digambarkan sebagai graf berbentuk pohon dimana akar dari pohon ini adalah folder /root. Di dalam folder root, terdapat folder-folder lain yang berisi file di dalamnya. Jika ditotal, folder yang ada hampir berkisar 200 folder dan file yang ada sekitar 3000 file. Secara kasar, ilustrasi file sistem Android dalam graf yang sangat sederhana tampak pada gambar di bawah



Gambar 3 Ilustrasi File Sistem Android dalam Bentuk Graf Sederhana

Berikut adalah ilustrasi dari ide untuk memastikan keabsahan nandroid backup.



Gambar 4 Flow Data dalam Proses Backup dan Restore

Dalam melakukan proses backup, *custom recovery* terlebih dahulu melakukan langkah 1, yaitu mendapatkan list folder dan file secara keseluruhan. List folder dan file didapat dengan melakukan pencarian dengan algoritma DFS. List folder dan file tersusun berdasarkan waktu dikunjunginya. Lalu list foler dan file yang telah didapat disimpan dalam suatu file teks. File teks ini nantinya akan dicocokkan dengan file berkas hasil *backup*. Setelah proses langkah 1 selesai, maka dilakukan langkah 2, yaitu proses kompresi file sistem menjadi suatu file berkas (dengan ekstensi .zip misalnya).

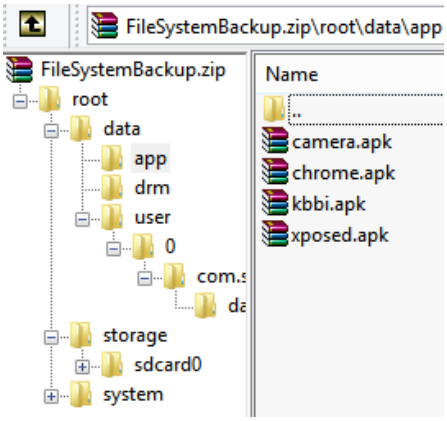
Dalam melakukan proses *restore*, *custom recovery* terlebih dahulu melakukan langkah 3, yaitu mengekstrak file berkas hasil *backup* ke sebuah folder, misal folder *temp*. Setelah berhasil diekstrak, lalu gunakan algoritma DFS dan pencocokan string untuk melakukan langkah 4. Algoritma DFS akan digunakan untuk mendapatkan nama tiap folder dan file yang ada di dalam folder temp tadi. Setiap kali didapat suatu nama folder/file, maka cocokkan dengan teks yang ada di file teks yang telah dibuat ketika proses *backup*. Lakukan proses ini berulang-ulang, apabila terdapat perbandingan *string* yang tidak cocok, itu berarti telah ada modifikasi terhadap file berkas hasil *backup* sehingga proses *restore* dibatalkan. Apabila semua perbandingan *string* cocok, maka proses restore bisa dilakukan.

Mari kita ilustrasikan proses *backup* dan *restore* ini menggunakan pseudo file sistem yang tergambar pada pada Gambar 3. Ketika melakukan *backup*, langkah-langkah yang harus dilakukan adalah sebagai berikut:

1. Dari langkah 1 pada Gambar 4: membuat list folder dan file, didapat file teks sebagai berikut (dimana nama folder diawalai karakter *slash)*.

```
/root
/data
/app
camera.apk
chrome.apk
kbbi.apk
xposed.apk
/drm
/user
/0
/com.sony.snei.vu
/databases
app.db
/storage
/sdcard0
/DCIM
/image
/music
/video
/system
/app
contacs.apk
gmail.apk
music.apk
/bin
mount
netstat
su
/etc
/bluetooth
/init.d
/wifi
wifi.conf
/framework
service.jar
settings.jar
```

2. Langkah selanjutnya adalah membuat file berkas yang merupakan hasil salinan dari file sistem. Kita asumsikan bahwa file telah berhasil dibuat dengan nama FileSystemBackup.zip.

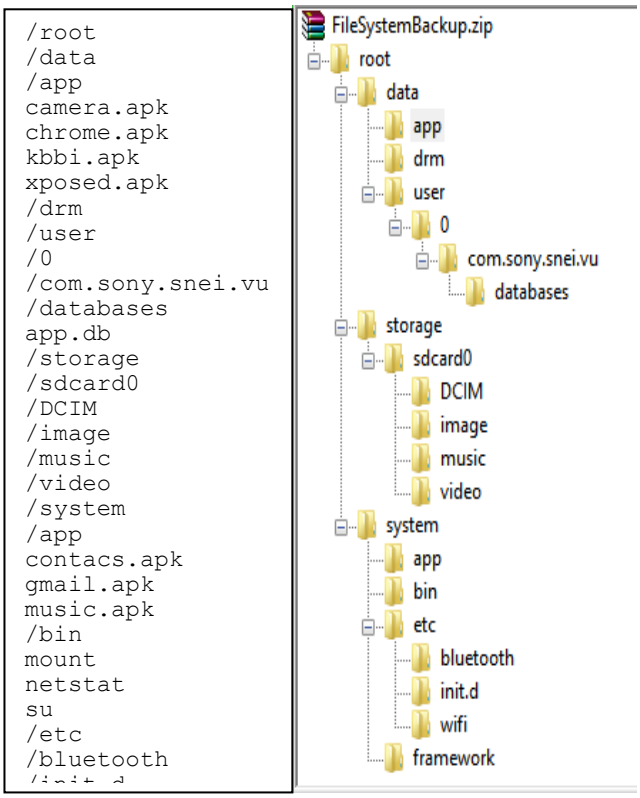


Gambar 5 Ilustrasi File Hasil Backup

3. Jika file teks telah terbentuk dan file berkas hasil *backup* telah selesai dikompresi, maka proses *backup* telah selesai.

Sedangkarn berikut adalah prosedur yang harus dijalankan ketika proses restore:

1. Ekstrak terlebih dahulu FileSystemBackup.zip ke suatu folder, misal folder **temp**. Itu berarti, di dalam folder **temp** ada folder **root** yang merupakan akar dari file sistem yang telah di-*backup* sebelumnya
2. Dengan menggunakan algoritma DFS, dapatkan nama tiap folder atau file yang ada di folder **temp** dan cocokkan dengan file teks yang didapat dari langkah 1 pada Gambar 4.



Gambar 6 Ilustrasi Perbandingan File Teks dan File Hasil Backup

Dari Gambar 6, berikut adalah pencocokan *string* yang terjadi:

- Pencocokan I : **/root** dengan **root**
- Pencocokan II: **/data** dengan **data**
- Pencocokan III: **/app** dengan **app, dst**

3. Apabila semua perbandingan *string* menghasikan nilai *true*, maka proses *restore* bisa dilakukan.

Berikut adalah algoritma dari fungsi *backup*.

```
procedure logBackup(input v:path folder)

{ Mengunjungi seluruh folder dan dan
membaca seluruh file yang ada
Masukan: folder /root
Keluaran: semua nama folder dan file yang
tersusun berdasarkan waktu kunjungannya
dan sebuah file berkas berisi seluruh
folder }

Deklarasi
       w : path folder
Algoritma
       bacaFolder (v)
       bacaFile (v)
       dikunjungi[v] ← true
       for tiap path w yang bertetangga
dengan path v do
              if not dikunjungi[w] then
                     logBackup(w)
              endif
       endfor


procedure bacaFolder (input v:path folder)

{ Membaca nama folder dan meng-append-nya
ke file teks
Masukan: path folder
Keluaran: file teks ter-append oleh nama
folder yang menjadi masukan }

Deklarasi
       F : file teks
Algoritma
       append_text(f, v)


procedure bacaFile (input v:path folder)

{ Membaca seluruh file yang ada di suatu
folder dan memasukkan datanya ke file teks
Masukan: path folder
Keluaran: file teks berhasil di-append
list file yang ada di folder yang menjadi
masukan }

Deklarasi
       ft : file teks
Algoritma
       for tiap file f yang ada di path v
do
              append_text(ft, f)
       endfor
```

Prosedur **logBackup** di atas adalah adalah prosedur yang memanfaatkan algoritma DFS. Masukan awal dari prosedur **logBackup** adalah *path* dari folder **/root**. Setelah itu, proses rekursif akan dilakukan dimana prosedur **logBackup** akan menerima input masukan *path* folder-

folder yang di dalam folder **/root**. Setiap kali pemanggian prosedur, akan dilakukan pembacaan nama file yang ada di folder masukan. Semua folder dan file yang dibaca dimasukkan ke dalam file teks.

Sedangkan berikut adalah algoritma dari fungsi *restore*:

```
function checkRestore(input v: path
folder, f: file teks) → boolean

{ Membaca seluruh folder dan file yang
ada dan menyalinnya ke folder /root
Masukan: path folder
Keluaran: true/false. Apakah file
hasil backup berbeda dengan hasil log
yang telah dibuat }

Deklarasi
       s0 : string
       s1 : string
       s2 : array of string
       w : path folder

Algoritma
       s1 ← getNamaFolder(v)
       s0 ← getNextStringFromTeks(f)
       if not isKataSama(s0,s1) then
              return false

       endif
       s2 = getNamaFiles(v)
       for string s in array of string
s2 do
          s0 ← getNextStringFromTeks(f)
          if not isKataSama(s,s0) then
               return false
          endif
       endfor
       Dikunjungi[v] ← true
       for tiap path w yang
bertetangga dengan path v do
           if not dikunjungi[w] then
               if not checkRestore(w)
                    return false
               endif
           endif
       endfor
       return true


function isKataSama (input s1:string,
s2:string) → boolean

{ Membandingkan 2 buah string
Masukan: 2 buah string
Keluaran: boolean, nilai true berarti
kedua kata sama sedangkan nilai false
berarti kedua kata berbeda }

Deklarasi
       counter : integer
Algoritma
       if s1.length > s2.length then
          counter ← s1.length
       else
          counter ← s2.length
       endif
       for i ← 1 to s1.length do
          if not s1[i] = s2[i]
             return false
          endif
       endfor
```

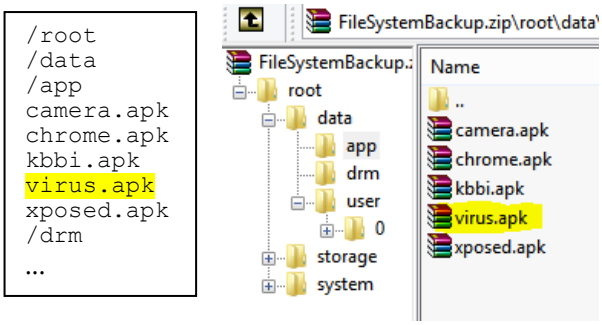
Fungsi **checkRestore** di atas adalah fungsi yang akan mencocokkan isi dari folder **temp** dengan file teks yang telah dibuat pada saat proses *restore* berlangsung. Keluaran dari fungsi tersebut adalah sebuah *boolean* dimana nilai *true* menunjukkan bahwa file hasil *backup* integritasnya teruji (tidak ada perubahan) sedangkan nilai *false* menunjukkan terjadinya ketidakcocokkan antara file hasil *backup* dengan file teks *log* yang dibuat saat proses *backup*.

Fungsi **isKataSama** adalah fungsi yang menerima masukan dua buah *string* dan mengimplementasikan konsep *pattern matching*. Langkahnya adalah pertama, cari terlebih dahulu teks mana yang memiliki karakter lebih panjang. *String* yang memiliki karakter lebih panjang kita anggap sebagai teks dan string yang memiliki karakter lebih pendek kita anggap sebagai *pattern*. Lalu bandingkan karakter demi karakter, apabila ditemukan satu karakter yang berbeda, maka dapat disimpulkan bahwa kedua *string* tadi berbeda.

IV. ANALISIS

Dalam konteks yang sederhana, penggunaan algoritma DFS dan pencocokan *string* untuk menguji keabsahan nandroid backup ini cukup berhasil. Pencocokan string berhasil mendeteksi perbedaan antara dua kata yang berbeda. Algoritma DFS juga mampu mengeksplorasi folder root secara terurut. Dengan demikian program yang penulis buat, mampu mendeteksi adanya perbedaan pada file berkas hasil *backup* dengan file teks yang merupakan file log yang berfungsi sebagai sumber file untuk memvalidasi bahwa file berkas hasil berkas hasil *backup*.

Namun demikian, cara ini masih mampu ditembus dengan mudah. Contohnya adalah ketika file hasil *backup* disisipkan program jahat dengan syarat file teks juga turut diubah:



Gambar 7 Dengan Metode Ini, virus.apk Tidak Dapat Dideteksi oleh Aplikasi yang Penulis Buat

Oleh karena itu, diperlukan metode lain untuk menjamin bahwa file teks dan file hasil *backup* tidak mengalami proses pengeditan. Salah satu cara paling mudah adalah dengan melibatkan enkripsi terhadap file teks yang dibuat. Metode enkripsi yang umum digunakan salah satunya adalah 3DES (Triple Data Encryption Standard) yang merupakan salah satu bentuk enkripsi simetris. Berikut

adalah perbandingan file teks yang berbentuk plain teks (berada di *textbox* sebelah kiri) dan teks hasil enkripsi dengan *passphrase* "strategi_algoritma" (*textbox* sebelah kanan):

```
/root
/data
/app
camera.apk
chrome.apk
kbbi.apk
xposed.apk
/drm
/user
/0
/com.sony.snei.vu
/databases
app.db
/storage
/sdcard0
/DCIM
/image
/music
/video
/system
/app
contacs.apk
gmail.apk
music.apk
/bin
mount
netstat
su
/etc
/bluetooth
/init.d
/wifi
wifi.conf
/framework
service.jar
settings.jar
```

```
XiKFT3l+lRto63
kxcTESk8gKAfWv
34HC3BriVrfd96
62aDTkvOcPi0Qo
aMbjas0NDC78cD
XTwj07ShBPS68Q
XxAC+X+3h3CG/H
wrPMJLuBPDyQLC
+CjWMLWCeeeYP9
6vHKvpTIiXZIiF
ne4LX1XKlUr0I/
IxglaZCYEqrB7E
BHhm8jqGupGw3X
A7pPrBqmPpWFQd
TprB/waKtuVLus
ITbfru7jkytHq3
KlzHwss9oi+RWH
2iXRPBk497Wopg
40Gli0WUmh1GV3
hzlmT1rGm/k3yv
BjYrqReZC7rsmB
xRHVvMyqwvv5Fb
0oJ2YpPahcBwMr
R/ADq0buHZIc/U
D9qh/Cx7D8+F6+
iKJv39NNmx7adS
0xGTTRfb1uNI3n
xXoXQkh6TIM8I6
spFUpO/N8NxaPA
Hsmny5686qPiEa
GE6xW2TnVWwttD
bhzc93Qw1GMxb0
```

Dengan tambahan metode enkripsi pada file teks, maka probabilitas keberhasilan memasukkan *unwanted program* ke dalam file *backup* akan semakin kecil. Karena meskipun file berbahaya berhasil dimasukkan ke dalam file berkas hasil *backup*, tetapi untuk mengubah file teksnya tentu tidaklah mudah karena *passphrase*-nya dirahasiakan. Hal yang terjadi jika file berkasi hasil *backup* disusupi program jahat dan file teks tetap tidak diubah adalah ketika proses verifikasi, fungsi checkRestore akan mengembalikan nilai *false* sehingga proses *restore* tidak dapat dilakukan.

V. KESIMPULAN

Kesimpulan yang didapat setelah melakukan analisis dari permasalahan ini adalah sebagai berikut:

- Algoritma DFS sangat cocok diimplementasikan untuk meng-crawl suatu file sistem, dalam hal ini adalah file sistem Android.
- Algoritma DFS dan pencocokan string dapat digunakan untuk memverifikasi proses nandroid backup and restore.
- Tambahan keilmuan lain seperti enkripsi maupun kriptografi mampu meningkatkan keamanan terhadap file nandroid *backup and restore*.

REFERENSI

[1] http://www.phonedog.com/2011/05/12/what-are-custom-recoveries-for-android/ Diakses tanggal 19 Desember 2013

[2] http://cikalinspirasi.blogspot.com/2013/05/algoritma-dept-first-search-dfs.html Diakses tanggal 19 Desember 2013

[3] Munir, Rinaldi., 2009, Diktat Kuliah IF2211 – Strategi Algoritma., Institut Teknologi Bandung.

[4] http://journal.uii.ac.id/index.php/Snati/article/viewFile/1413/1193 Diakses tanggal 19 Desember 2013

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, 20 Desember 2013

Asep Saepudin – 13511093