IMPLEMENTASI GPU PROCESSING SYSTEM UNTUK DECHIPERING KODE TERENKRIPSI

IMPLEMENTATION OF GPU PROCESSING SYSTEM FOR CODE ENCRYPTED DECHIPERING

Howard Aiken Siburian, Henry Rossi Andrian, S.T., M.T., Mochammad Fahru Rizal, S.T., M.T.
Program Studi D3 Teknik Komputer, Fakultas Ilmu Terapan, Universitas Telkom
howard.aiken88@gmail.com

Abstrak

Sistem komunikasi telah mengalami perkembangan yang cukup pesat dengan banyaknya teknologi telekomunikasi yang bermunculan. *Global Sistem for Mobile Communication* (GSM) adalah salah satu teknologi komunikasi yang telah dipakai hampir di seluruh dunia. Makin bertambahnya jumlah pengguna membuat keamanan informasi yang dipertukarkan menjadi salah satu aspek yang cukup penting untuk diperhatikan. Keamanan informasi pada jaringan GSM dapat dianalisa menggunakan *GPU Processing System*. Selain itu, *GPU Processing System* dapat digunakan untuk membantu proses komputasi yang akan dilakukan. GPU *Processing System* memiliki fungsi untuk melakukan suatu pekerjaan yang membutuhkan banyak perhitungan dalam waktu singkat. Berdasarkan hasil implementasi, GPU *Processing System* dalam proyek akhir ini mampu mengurangi proses komputasi yang lama menjadi cepat.

Abstract

The communication system has developed quite rapidly with the many emerging telecommunication technologies. Global Systems for Mobile Communications (GSM) is one of the most used communication technologies worldwide. The increasing number of users make the secured information for the exchanged is one of the important aspects to note.Information on GSM networks can be analyzed using GPU Processing System. In addition, GPU Processing System can be used to assist the computing process that will be done. GPU Processing System has a function to perform a job that requires a lot of calculations in a short time. Based on the implementation, GPU Processing System in the final process is able to reduce the slow computing more faster than the old version.

1. Pendahuluan

1.1 Latar Belakang

Sistem komunikasi telah mengalami perkembangan yang cukup pesat dengan banyaknya teknologi telekomunikasi yang bermunculan. *Global Sistem for Mobile Communication* (GSM) adalah salah satu teknologi dibidang komunikasi yang telah dipakai hampir di seluruh dunia, namun jaringan tersebut dilengkapi oleh sistem keamanan yang masih lemah. Sistem keamanan saat ini menggunakan sistem enkripsi *A5/1*. Seiring berjalannya waktu kasus peretasan terhadap jaringan tersebut untuk kebutuhan mengambil data-data tertentu yang dimiliki oleh target namun ada juga yang bertujuan untuk menghancurkan data atau sistem yang berdampak terhadap kerusakan digital. Algoritma A5/1 memiliki kelemahan serius didalam keamanan komunikasi datanya. Oleh karena itu, akan dibuat suatu proyek akhir yang mempunyai tujuan untuk melakukan analisis kelemahan dari keamanan algoritma A5/1 yang digunakan oleh jaringan GSM. Data terenkripsi yang telah didapat akan dicari kunci chiper-nya dengan menggunakan program bernama *Kraken* yang dijalankan menggunakan *GPU (Graphic Processing Unit)* yang tersambung dengan *Harddisk* berisi *Rainbow Table*. Cara ini digunakan untuk menganalisa adanya kelemahan keamanan pada data komunikasi GSM dengan menggunakan *GPU Processing System* yang akan membantu proses komputasi yang dilakukan.

1.2 Rumusan Masalah

Beberapa rumusan masalah dalam penyusunan Proyek Akhir ini adalah sebagai berikut.

1. Bagaimana cara implementasi *GPU Processing System* pada Ubuntu 14.04 LTS?
2. Bagaimana cara pengujian *GPU Processing System* untuk dekripsi data?

1.3 Tujuan

Berdasarkan rumusan masalah diatas maka diambil beberapa tujuan dari penyusunan Proyek Akhir ini sebagai berikut :

1. Dapat mengimplementasikan *GPU Processing System* pada Ubuntu 14.04 LTS.
2. Dapat mengetahui cara pengujian *GPU Processing System* untuk dekripsi data.

1.4 Batasan Masalah

Adapun batasan masalah dari Proyek Akhir ini adalah :

1. Hanya membahas algoritma A5/1.

2. Hanya membahas tentang keamanan data komunikasi GSM.
3. Tidak membahas lebih jauh tantang dekripsi data komunikasi GSM.
4. Tidak membahas lebih jauh tentang *GPU Processing System.*

1.5 Definisi Operasional

Terdapat beberapa definisi operasional yang ada dalam sistem yang akan dibangun yaitu:

1. GSM *(Global System for Mobile Communication)*
   Global System for Mobile Communication (GSM) awalnya berasal dari singkatan *Grupe Special Mobile* yang memiliki pengertian sebuah teknologi komunikasi seluler yang bersifat digital.
2. GPU *(Graphic Processing Unit)*
   GPU adalah sebuah CPU tambahan yang bertugas untuk melakukan pekerjaan yang membutuhkan banyak perhitungan dalam satu waktu, salah satunya adalah pengolahan grafis.
3. ENKRIPSI
   Enkripsi adalah proses *encoding* (pengkodean/penyandian) sebuah pesan, dan proses tersebut bisa mengambil berbagai macam bentuk.
4. DEKRIPSI
   Dekripsi adalah proses untuk mengubah *chiperteks* menjadi *plainteks* atau pesan asli.

1.6 Metode Pengerjaan

Metode yang digunakan dalam menyusun Proyek Akhir ini adalah :

1. Studi literatur
   Pencarian referensi dan sumber – sumber untuk mempelajari konsep dan teori yang berkaitan dengan GSM dan enkripsi pada GSM. Hal ini dilakukan sebagai landasan untuk analisis kebutuhan sistem dan implementasi sistem yang akan dibangun.
2. Analisis Kebutuhan Sistem
   Landasan konsep dan teori yang telah dilakukan pada tahap studi literatur digunakan untuk menganalisis kebutuhan sistem kemudian mengimplementasikan sistem dari hasil analisis kebutuhan ini.
3. Implementasi dan Pengujian Sistem
   Untuk implementasi sistem yang akan dibangun, dilakukan dengan instalasi perangkat – perangkat lunak pada sistem. Kemudian dilakukan pengujian dari implementasi sistem yang sebelumnya telah dilakukan. Pada tahapan ini dilakukan pengujian *rainbow table*, XoR *keystream* dengan *encrypted* burst, *crack* XoR'ed burst dan pencarian *chiper key* pada Ubuntu 14.04 LTS. Hal ini dilakukan untuk memastikan sistem yang telah dibuat dapat berjalan dengan baik.
4. Pembuatan Laporan
   Pada tahap terakhir ini, dilakukan dokumentasi dan penyusunan laporan dari semua proses tahapan yang telah dilakukan.

2. Tinjau Pustaka

2.1 GSM (*Global System for Mobile Communication*)

Global System for Mobile Communication (GSM) awalnya berasal dari singkatan *Grupe Special Mobile* yang memiliki pengertian sebuah teknologi komunikasi seluler yang bersifat digital. Teknologi GSM banyak diterapkan pada komunikasi bergerak, khususnya telepon genggam. Teknologi ini memanfaatkan gelombang mikro dan pengiriman sinyal yang dibagi berdasarkan waktu, sehingga sinyal informasi yang dikirim akan sampai pada tujuan. GSM dijadikan standar global untuk komunikasi seluler sekaligus sebagai teknologi yang paling banyak digunakan orang di seluruh dunia.

2.2 Ubuntu 14.04 LTS

Ubuntu Versi 14.04 *"Trusty Tahr"* merupakan distribusi Linux yang paling populer menggunakan *user interface* Unity yang khas dan disesuaikan. Trusty Tahr merupakan edisi dengan dukungan jangka panjang *"Long Term Support"* (LTS) selama 5 tahun, berupa dukungan keamanan berikut jalur upgrade yang lebih mudah dibandingkan rilis versi LTS (12.04) sebelumnya.

2.3 *Graphics Processing Unit* (GPU)

Graphics Processing Unit (GPU) adalah *chip* khusus pada sebuah *Video Graphics Array* (VGA) yang dirancang untuk memanipulasi dan mengubah memori sehingga mempercepat pembangunan grafis, namun pada beberapa tahun terakhir GPU juga digunakan untuk melakukan komputasi paralel[6] . GPU mempunyai core yang lebih banyak daripada *processor* CPU sehingga kini dimanfaatkan untuk pekerjaan komputasi yang berat, selain untuk game dan simulasi.Komputasi paralel dengan GPU merupakan suatu metode komputasi yang masih baru. Pada awalnya komputasi paralel dapat dilakukan dengan menggunakan *grid computing*, di mana suatu pekerjaan didistribusikan ke banyak komputer yang saling terhubung melalui jaringan. GPU memiliki sejumlah *multi-core processor* dan setiap *processor* memiliki SIMD- *processor* (*Single Instruction Multiple Data*) [4]. SIMD artinya dengan satu instruksi dapat mengeksekusi sejumlah data paralel dalam waktu yang bersamaan. GPU kini sangat efisien dalam memanipulasi grafis komputer dan struktur kinerja paralel yang membuatnya lebih efektif dibandingkan pengerjaan menggunakan CPU untuk penerapan suatu algoritma, karena pengolahan blok besar data dilakukan secara paralel atau terpartisi.

2.4 Kraken

Kraken adalah suatu *software open source* yang digunakan untuk mencari Kc dari algoritma A5/1 yang ada pada jaringan GSM. *Software* ini dikenal dengan performanya yang lebih efisien dan lebih cepat dalam mencari kunci chiper yang cocok untuk membuka data yang terenkripsi berupa bit dengan format txt.

2.5 Rainbow table

Rainbow table adalah *Precomputed table* yang digunakan untuk mengembalikan fungsi kriptografi *hash*. Umumnya digunakan untuk *crack hash* kata sandi. Idenya disini adalah hanya menyimpan bagian dari *precomputing tables* lalu bagian tabel lainnya akan dilakukan komputasi ulang pada saat pencarian.

2.6 Algoritma A5/1

A5/1 adalah algoritma enkripsi yang digunakan oleh sekitar 130 juta pelanggan GSM di Eropa untuk melindungi privasi *over-the-air* dari seluler suara dan data komunikasi mereka. Serangan yang terbaik diterbitkan terhadap itu membutuhkan antara 240 dan 245 langkah. Tingkat keamanan membuatnya rentan terhadap serangan berbasis *hardware* oleh organisasi besar, tetapi tidak untuk serangan berbasis *software* pada beberapa sasaran oleh hacker.

2.7 Kunci Chiper

Kunci Chiper (Kc) adalah kunci yang digunakan untuk enkripsi dan dekripsi. Sistem ini mengharuskan dua pihak yang berkomunikasi menyepakati suatu kunci rahasia yang sama sebelum keduanya saling berkomunikasi.

2.8 Hard Disk Drive

HDD atau *hard drive* adalah sebuah komponen perangkat keras yang menyimpan data sekunder dan berisi piringan magnetis. *Hard Disk Drive* diciptakan pertama kali oleh insinyur IBM, Reynold Johnson pada tahun 1956. Hard Disk Drive pertama tersebut terdiri dari 50 piringan berukuran 2 kaki (0,6 meter) dengan kecepatan rotasinya mencapai 1.200 rpm *(rotation per menute)* dengan kapasitas penyimpanan 4,4 MB.

2.9 Git

Git adalah perangkat lunak pengontrol versi atau proyek manajemen kode perangakat lunak yang diciptakan oleh Linus Torvalds, yang pada awalnya ditujukan untuk pengembangan kernel linux.

2.10 SDR *(Software Defined Radio)*

Software Defined Radio (SDR) ada yang menyebut juga *software radio* (SWR) diperkenalkan pertama kali pada tahun 1991 oleh joseph mitola istilah SDR ini digunakan untuk menujuk sebuah kelas radio yang dapat dikonfigurasikan ulang diprogam ulang, sehingga menghasilkan sebuah jenis perangkat komunikasi nirkabel dengan mode dan bend frekuensi ditentukan oleh fungsi perangkat lunak . SDR memiliki keuntungan karena sifat fleksibelitas (*flexsibeltiy*) ,lengkap dan dapat dikonfigurasikan ulang secara mudah (*complete and easy reconfigurasiability*) dapat disekala, dapat diperogam ulang (*reprogrammability*) secara dapat diperluas (*expandability*).

2.11 Pybombs

PyBOMBS (*Python Build Overlay Managed Bundle System*) adalah sistem manajemen baru yang digunakan untuk proses installasi GNURadio agar dapat berjalan dengan baik. GNURadio dasarnya menggunakan bahasa pemrograman python. Tujuan utama pybombs adalah untuk menggabungkan beberapa aplikasi yang digunakan untuk menyelesaikan suatu projek dengan menggunakan bahasa pemrograman python. Maka pybombs merupakan sistem dasar yang harus ada sebelum menggunakan GNURadio.

2.12 Kalibrate

Kalibrate adalah program Linux yang dapat memindai BTS GSM pada pita frekuensi yang diberikan dan dapat menggunakan BTS GSM untuk menghitung keseimbangan frekuensi osilator lokal.

2.13 Enkripsi

Enkripsi adalah proses yang dilakukan untuk mengamankan sebuah pesan (yang disebut *plaintext*) menjadi pesan yang tersembunyi (disebut *ciphertext*) adalah enkripsi (*encryption*). *Ciphertext* adalah pesan yang sudah tidak dapat dibaca dengan mudah.

2.14 GNU Radio

GNU Radio adalah perangkat lunak untuk membangun dan menyebarkan perangkat lunak sistem radio. Kerangka GNU Radio menyediakan pemrosesan sinyal yang panjang dan pengolahan blok untuk berkomunikasi dengan perangkat keras. GNU Radio telah digunakan array besar aplikasi radio dunia nyata, termasuk pengolahan audio, komunikasi ponsel, pelacakan satelit, sistem radar, jaringan GSM, dll.

2.15 RTL-SDR

RTL-SDR adalah sebuah usb dvb-t yang digunakan untuk menangkap siaran televisi digital. Alat ini hanya digunakan untuk streaming siaran televisi digital saja, namun bisa digunakan menjadi alat penerima *multi-mode* dan *multi-band* atau sebut saja alat yang bisa digunakan sebagai *hardware SDR*. Chipset DVB-T RTL-SDR adalah Realtek RTL2832U yang mampu menangkap *signal radio* dan frekuensi tertentu dan paket tersebut dalam bentuk RAW data, tentunya untuk dapat menterjemahkan RAW data ke dalam computer maka diperlukan *software* yang mampu melakukan proses decoding tersebut seperti GNURadio.

2.16 Pip
Pip adalah sistem manajemen paket yang digunakan untuk menginstal dan mengelola paket perangkat lunak yang ditulis dengan python.

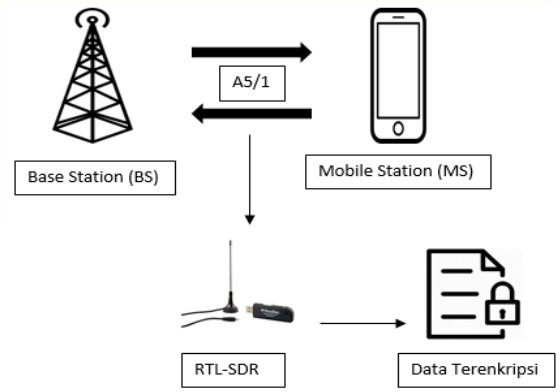
2.17 Gr-GSM
Gr-GSM adalah proyek yang berdasarkan penerima GSM yang ditulis oleh Piotr Krysik (juga penulis utama gr-gsm) untuk proyek Airprobe. Tujuannya adalah untuk menyediakan seperangkat alat untuk menerima informasi yang dikirimkan oleh peralatan / perangkat GSM.

2.18 Airprobe
Airprobe adalah salah satu proyek dari Gr-GSM. Airprobe merupakan aplikasi pendukung alat RTLSDR untuk *decoding* GSM. Dengan aplikasi ini dengan mudah untuk *decoding* sistem sms GSM.

2.19 GSM Framecoder
GSM Framecoder adalah aplikasi untuk menghitung *bursts* yang dihasilkan pada paket GSM untuk dapat di *decoding* datanya.

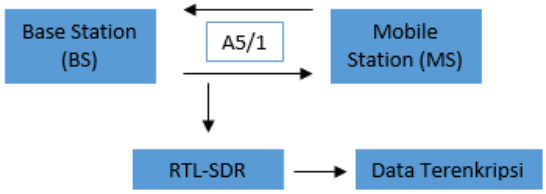
2.20 Wireshark
Wireshark adalah alat analisis jaringan yang sebelumnya dikenal dengan *Ethereal*. Wireshark dapat menangkap paket secara *real time* dan menampilkannya dalam format yang mudah dibaca manusia. Pada dasarnya, wireshark adalah penganalisis paket jaringan yang memberikan rincian singkat tentang protokol jaringan, dekripsi, informasi paket, dll. Wireshark adalah sumber terbuka dan dapat digunakan di Linux, Windows, OS X, Solaris, NetBSD, FreeBSD, dan banyak lainnya. Sistem informasi yang diambil melalui alat ini bisa dilihat melalui GUI atau TTY mode TShark Utility.

3. Analisis dan Perancangan
3.1 Analisis
Berikut ini adalah pembahasan mengenai analisis sistem yang sedang berjalan dan analisis sistem yang akan dibangun.

3.1.1 Gambaran Sistem Saat Ini


Gambar 3 - 1 Gambaran Sistem Saat Ini

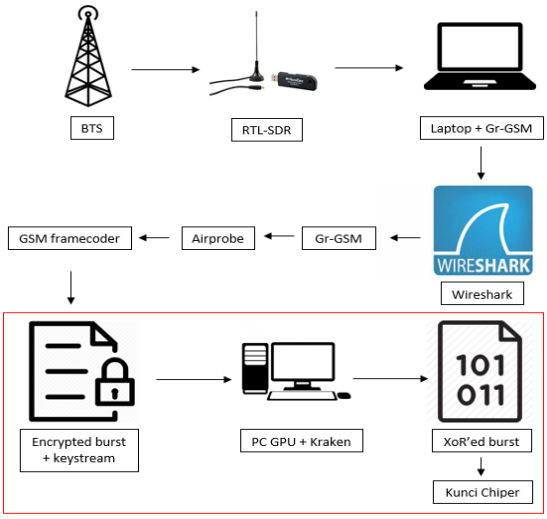
Saat ini, hal yang dapat dianalisa pada data komunikasi GSM yang dilindungi enkripsi A5/1 sangat terbatas. Hanya bisa mengambil data berupa *encrypted burst* dengan format .txt dari *Base Transceiver Station* dengan menggunakan RTL-SDR.

3.1.2 Blok Diagram Sistem Saat Ini


Gambar 3 - 2 Blok Diagram Sistem Saat Ini

3.1.3 Cara Kerja Sistem
Pada sistem saat ini enkripsi data komunikasi GSM antara *Base Station (BS)* dan *Mobile Station (MS)* menggunakan algoritma A5/1, pengambilan data terenkripsi menggunakan RTL-SDR, data terenkripsi yang telah didapat akan di*decoding* untuk mendapatkan *encrypted burst* berbentuk *bitstream*.

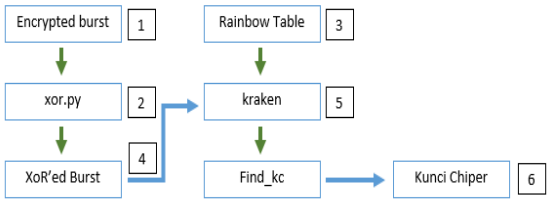
3.2 Perancangan Sistem
Di bawah ini adalah pemaparan gambaran sistem usulan.

3.2.1 Gambaran Sistem Usulan


Gambar 3 - 3 Gambaran Sistem Usulan

Gambaran sistem usulan ini adalah pengembangan dari sistem sebelumnya. Cara kerjanya ialah *encrypted burst* dan *keystream* yang telah didapatkan akan diproses menggunakan tool xor.py pada program kraken untuk mendapatkan xor'ed burst. Burst tersebut akan di crack dan didapatkan data indexes yang menunjukkan kemungkinan kunci chipernya.

3.2.2 Blok Diagram Sistem Usulan



Gambar 3 - 4 Blok Diagram Sistem Usulan

3.2.3 Cara Kerja

1. Menggunakan tool xor.py dengan menambahkan bit *keystream* dan bit *encrypted burst*.
2. Mendapatkan hasil pertambahannya yaitu xor'ed burst yang nantinya akan di *crack* ke dalam tool kraken.
3. Menghubungkan rainbow table berbentuk raw yang telah di *convert* dengan file indexes dengan perintah ./kraken ../indexes. Perintah ini dilakukan agar dapat melakukan *crack xor'ed burst*.
4. Salin *xor'ed burst* yang telah didapat dan *crack burst* tersebut dengan perintah *crack* untuk mendapatkan data index yang akan menunjukkan kunci chiper dari burst tersebut.
5. Salin hasil data index dari *xor'ed burst* tersebut ke dalam tool *find_kc* dan masukkan juga *frame number* dari data index tersebut untuk mendapatkan *potential chiper key*.
6. Tambahkan *frame number* urutan sebelumnya dan *xor'ed burst*nya untuk mengeliminasi Kc yang salah dan mendapatkan Kc yang sesuai.

3.2.4 Spesifikasi Sistem

Spesifikasi sistem yang digunakan pada proyek akhir ini terbagi menjadi dua yaitu perangkat keras (*hardware)* dan perangkat lunak (*software).*

3.2.4.1 Perangkat Keras

Perangkat keras merupakan komponen yang terlihat secara fisik, demikian perangkat keras yang digunakan dapat dilihat pada tabel 3-1 berikut.

Tabel 3 - 1 Perangkat Keras

No.	Spesifikasi minimum perangkat	Keterangan
1.	Processor AMD Phenom(tm) II X2 555 Processor x 2	Untuk memproses pencarian Kc.
2.	VGA Card Radeon HD 6570 DDR3 2Gb	Digunakan sebagai pendukung program kraken dan membantu proses komputasi yang dilakukan.
3.	Harddisk 2 Terrabyte	Digunakan sebagai tempat penyimpanan *rainbow table*.

3.2.4.2 Perangkat Lunak

Spesifikasi sistem *software* yang digunakan dalam perancangan ini terdapat pada tabel 3-2 berikut.

Tabel 3 - 2 Perangkat Lunak

No.	Perangkat Lunak	Keterangan
1.	Ubuntu 14.04 LTS	Sebagai sistem operasi.
2.	Kraken	Untuk mencari kunci chiper data komunikasi GSM.
3.	Git	Untuk menghubungkan library yang dibutuhkan software.
4.	Graphic driver	Untuk mengontrol VGA Card yang digunakan.
5.	AMDAPP SDK	Untuk memproses GPU berjalan sesuai sistem CPU.
6.	Calpp	Sebagai perpustakaan untuk penulisan ATI CAL.
7.	Pyrit	Untuk mendeteksi kekuatan perangkat yang terhubung.

4. Implementasi dan Pengujian

4.1 Implementasi

Implementasi bertujuan untuk menerapkan analisis dan perancangan yang telah dilakukan sehingga aplikasi dapat diinstalasi dan dikonfigurasikan. Berikut langkah-langkah instalasi aplikasi yang digunakan untuk proyek akhir ini :

4.1.1 Instalasi Graphic Driver pada Ubuntu 14.04 LTS

Berikut langkah-langkah melakukan instalasi Graphic Driver.

```
root@howard:~# ls fglrx
fglrx_15.201-0ubuntu1_amd64_UB_14.01.deb
fglrx-amdcccle_15.201-0ubuntu1_amd64_UB_14.01.deb
fglrx-core_15.201-0ubuntu1_amd64_UB_14.01.deb
fglrx-dev_15.201-0ubuntu1_amd64_UB_14.01.deb
```

Gambar 4 - 1 Driver yang telah didownload

1. Download driver VGA card.
2. Kemudian ketik dpkg –i fglrx*.deb untuk melakukan instalasi.
3. Kemudian ketik *aticonfig –initial* untuk melihat driver yang telah diinstal.
4. Kemudian ketik *fglrxinfo* untuk melihat info driver.

```
root@howard:~# fglrxinfo
display: :0  screen: 0
OpenGL vendor string: Advanced Micro Devices, Inc.
OpenGL renderer string: AMD Radeon HD 6500 Series
OpenGL version string: 4.5.13399 Compatibility Profile Context 15.201.1151
```

Gambar 4 - 2 fglrxinfo

4.1.2 Instalasi git pada Ubuntu 14.04 LTS

Perintah untuk instalasi git pada terminal :

1. apt-get install git

4.1.3 Instalasi AMDAPPSDK pada Ubuntu 14.04 LTS

Berikut langkah-langkah melakukan instalasi AMDAPPSDK.

1. Download AMDAPPSDK di website AMD.
2. Konfigurasi AMDAPPSDK, ketik *nano /root/.bashrc*.

```
 GNU nano 2.2.6            File: /root/.bashrc

# AMD APP SDK
export AMDAPPSDKROOT=/opt/AMDAPP
export AMDAPPSDKSAMPLESROOT=/opt/AMDAPP/
export LD_LIBRARY_PATH=${AMDAPPSDKROOT}lib/x86_64:${LD_LIBRARY_PATH}
export ATISTREAMSDKROOT=$AMDAPPSDKROOT
```

Gambar 4 - 3 Konfigurasi AMDAPPSDK

3. Kemudian ketik *mkdir amdappsdk* untuk membuat direktori baru.
4. Kemudian ketik *cp /home/howard/Downloads/AMD-APP-SDK-v2.9-lnx64.tgz /root/amdappsdk/* untuk copy file yang telah didownload.
5. Kemudian ketik *cd amdappsdk.*
6. Kemudian ketik *tar –xvf AMD-APP-SDK-v2.9-lnx64.tgz.*
7. Kemudian ketik *sudo ./Install-AMD-APP.sh*

```
root@howard:~/amdappsdk# ./Install-AMD-APP.sh
============================================================
64-bit Operating System Found..

Starting Installation of AMD APPSDK v2.9 ....
 SDK package name is :AMD-APP-SDK-v2.9-RC-lnx64.tgz
Checking Latest Version Info.....
Continuing in background, pid 3795.
......
 Continuing Installation...
============================================================
```

Gambar 4 - 4 Install AMDAPPSDK

4.1.3 Instalasi Cal++ pada Ubuntu 14.04 LTS

Berikut langkah-langkah melakukan instalasi Cal++.

1. Untuk copy file cal++ yang telah diunduh ke */root/* ketik cp */home/howard/Downloads/calpp-0.90.tar.gz /root/*
2. Ekstrak file cal++, ketik *tar –xvf calpp-0.90.tar.gz*
3. Masuk ke direktori calpp-0.90 ketik *cd calpp-0.90*
4. Konfigurasi Cal++ ketik *nano CmakeLists.txt*
5. Cek konfigurasi ketik *cmake* .
6. Instalasi Cal++ ketik *make install*

4.1.4 Instalasi Pyrit pada Ubuntu 14.04 LTS

Berikut langkah-langkah melakukan instalasi Pyrit.

1. Git clone pyrit ketik *git clone https://github.com/xiao106347/pyrit.git*
2. Masuk ke direktori Pyrit ketik *cd pyrit/pyrit*
3. Instalasi pyrit ketik *python setup.py install build*
4. Masuk ke direktori cpyrit_calpp ketik *cd .. cpyrit_calpp*
5. Instalasi Cal++ plugin ketik *python setup.py build install*
6. Menampilkan perangkat keras yang tersedia ketik *pyrit list_cores*

```
root@howard:~# pyrit list_cores
Pyrit 0.4.1-dev (C) 2008-2011 Lukas Lueg http://pyrit.googlecode.com
This code is distributed under the GNU General Public License v3+

The following cores seem available...
#1:  'CAL++ Device #1 'AMD GPU DEVICE''
#2:  'CPU-Core (SSE2)'
```

Gambar 4 - 5 Perangkat keras yang tersedia

7. Menampilkan kinerja perangkat keras yang tersedia ketik *pyrit benchmark*

```
root@howard:~# pyrit benchmark
Pyrit 0.4.1-dev (C) 2008-2011 Lukas Lueg http://pyrit.googlecode.com
This code is distributed under the GNU General Public License v3+

Running benchmark (16061.0 PMKs/s)... | ^[[3~[OR^[OR^[OQ^[OR

Computed 16060.98 PMKs/s total.
#1: 'CAL++ Device #1 'AMD GPU DEVICE'': 15633.7 PMKs/s (RTT 1.3)
#2: 'CPU-Core (SSE2)': 845.8 PMKs/s (RTT 3.0)
```

Gambar 4 - 6 Kinerja perangkat keras

4.1.5 Instalasi Program Kraken pada Ubuntu 14.04 LTS

Berikut langkah-langkah melakukan instalasi Kraken.

1. Pertama clone program, ketik *git clone* https://github.com/joswr1ght/kraken.git
2. Masuk ke direktori ketik *cd kraken*
3. Instalasi kraken menggunakan GPU ketik *make –C a5_ati*

4.1.6 Instalasi pip pada Ubuntu 14.04 LTS

Berikut langkah-langkah melakukan instalasi pip.

1. Untuk instalasi ketik apt-get install python-pip

```
root@howard:~# apt-get install python-pip
Reading package lists... Done
Building dependency tree
Reading state information... Done
python-pip is already the newest version.
0 upgraded, 0 newly installed, 0 to remove and 8 not upgraded.
```

Gambar 4 - 7 Instalasi pip

4.1.7 Instalasi PyBOMBS pada Ubuntu 14.04 LTS

Berikut langkah-langkah melakukan instalasi PyBOMBS.

1. Menghubungkan dengan repositori pybombs ketik *git clone https://github.com/pybombs/pybombs.git*
2. Instalasi pybombs dengan sistem manajemen pip ketik *pip install PyBOMBS*
3. Konfigurasi *prefix* pybombs pada direktori "/usr/local" ketik *sudo pybombs prefix init /usr/local/ -a default_prx*
4. Konfigurasi pybombs dengan *default prefix* ketik *sudo pybombs config default_prefix default_prx*
5. Menambahkan recipes untuk pybombs ketik *sudo pybombs recipes add gr-recipes git+https://github.com/gnuradio/gr-recipes.git* dan *sudo pybombs recipes add gr-etcetera git+https://github.com/gnuradio/gr-etcetera.git*

4.1.8 Instalasi Gr-GSM pada Ubuntu 14.04 LTS

Berikut langkah-langkah melakukan instalasi Gr-GSM.

1. Menghubungkan repositori Gr-GSM ketik *git clone https://github.com/ptrkrysik/gr-gsm.git*
2. Instal Gr-GSM dengan sistem manajemen paket pybombs ketik *pybombs install gr-gsm*

```
root@howard:~# pybombs install gr-gsm
PyBOMBS - INFO - PyBOMBS Version 2.3.2
PyBOMBS.install_manager - INFO - Phase 1: Creating install tree and installing b
inary packages:
PyBOMBS.install_manager - INFO - No packages to install.
```

Gambar 4 - 8 Instalasi Gr-GSM

4.1.9 Instalasi Kalibrate pada Ubuntu 14.04 LTS

Berikut langkah-langkah melakukan instalasi Kalibrate.

1. Menghubungkan repositori kalibrate ketik *git clone https://github.com/steve-m/kalibrate-rtl*
2. Masuk ke direktori kalibrate-rtl ketik *cd /kalibrate-rtl/*
3. Memuat program dengan perintah *"./bootstrap"* tujuan untuk memberitahu bahwa program akan dikonfigurasi pada komputer ketik *./bootstrap && CXXFLAGS='-W -Wall -03'*
4. Memeriksa kebutuhan sebelum membangun aplikasi ketik *./configure*
5. Membangun sumber kode sebelum melakukan instalasi ketik *make*
6. Melakukan instalasi dan memindahkan semua yang dibutuhkan aplikasi ke direktori ketik *make install*

4.1.10 Instalasi GNU Radio pada Ubuntu 14.04 LTS

Berikut langkah-langkah melakukan instalasi GNU Radio.

1. Install GNU Radio menggunakan sistem manajemen paket pybombs ketik *pybombs install gnuradio*

```
root@howard:~# pybombs install gnuradio
PyBOMBS - INFO - PyBOMBS Version 2.3.2
PyBOMBS.install_manager - INFO - Phase 1: Creating install tree and installing b
inary packages:
PyBOMBS.install_manager - INFO - No packages to install.
```

Gambar 4 - 9 Instalasi gnuradio

4.1.11 Instalasi Wireshark pada Ubuntu 14.04 LTS

Berikut langkah-langkah melakukan instalasi Wireshark.

1. Menambahkan repositori wireshark ketik *sudo add-apt-repository ppa:wireshark-dev/stable*
2. Perbaharui repositori wireshark ketik *apt-get update*
3. Instal aplikasi wireshark ketik *apt-get install wireshark*

```
root@howard:~# apt-get install wireshark
Reading package lists... Done
Building dependency tree
Reading state information... Done
wireshark is already the newest version.
0 upgraded, 0 newly installed, 0 to remove and 7 not upgraded.
```

Gambar 4 - 10 Instalasi Wireshark

4.1.12 Instalasi Airprobe pada Ubuntu 14.04 LTS

Berikut langkah-langkah melakukan instalasi Airprobe.

1. Menghubungkan dengan repositori airprobe ketik *git clone https://github.com/iamckn/airprobe*
2. Masuk ke direktori airprobe ketik *cd airprobe/*
3. Masuk ke direktori gsmdecode ketik *cd gsmdecode/*
4. Memuat program dengan konfigurasi *"bootstrap"* pada direktori gsmdecode ketik *./bootstrap*
5. Menjalankan konfigurasi *"configure"* pada direktori gsmdecode ketik *./configure*
6. Membangun sumber kode ketik *make*
7. Masuk ke direktori *"/gsm-receiver"* pada direktori *"/airprobe"* ketik *cd gsm-receiver/*
8. Memuat program dengan perintah "bootstrap" ketik *./bootstrap*
9. Menjalankan konfigurasi dan membangun tool yang dibutuhkan ketik *./configure && make -j4*

4.1.13 Instalasi GSM Framecoder pada Ubuntu 14.04 LTS

Berikut langkah-langkah melakukan instalasi GSM Framecoder.

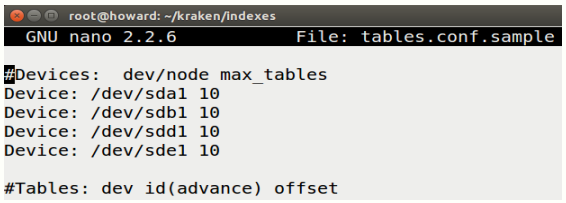
1. Mengambil file paket instalasi aplikasi pada server ketik *wget www.ks.uni-freiburg.de/download/misc/gsmframecoder.tar.gz*
2. Membuat direktori baru dengan nama *"gsmframecoder"* ketik *mkdir gsmframecoder*
3. Memindahkan file paket instalasi yang didapat ke direktori *"gsmframecoder"* ketik *mv gsmframecoder.tar.gz gsm gsmframecoder*
4. Masuk ke direktori *"gsmframecoder"* ketik *cd gsmframecoder/*
5. Mengekstrak file ketik *tar zxvf gsmframecoder.tar.gz*
6. Masuk ke direktori file yang telah diekstrak ketik *cd gsmframecoder*
7. Masuk ke direktori *"test"* ketik *cd test*
8. Mengkompilasi ulang file ketik *make clean*
9. Membangun sumber kode ketik *make*

4.1.14 Konfigurasi Indexes

Berikut langkah-langkah melakukan konfigurasi indexes.

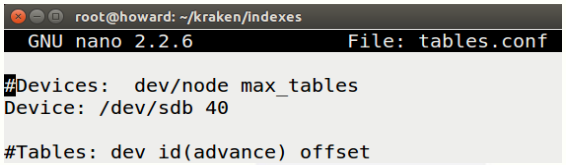
1. Konfigurasi pada file *tables.conf.sample* dengan masuk ke dalam direktori indexes yang ada pada kraken dengan perintah berikut pada terminal ketik *cd kraken/indexes*

2. Masuk ke dalam file *tables.conf.sample* dengan perintah berikut pada terminal ketik *nano tables.conf.sample*
3. Pada file *tables.conf.sample* terdapat contoh alokasi file rainbow table pada partisi. Pada gambar dibawah ini terdapat 4 alokasi partisi yaitu sda1, sdb1, sdd1 dan sde1. Masing-masing partisi akan dialokasikan 10 file hasil convert rainbow table.



Gambar 4 - 11 Alokasi partisi rainbow table

4. Pada kasus ini, alokasi pada partisi yaitu /dev/sdb 40. Save dengan nama table.conf agar tidak mengubah contoh alokasi tabel yang ada pada program kraken. Dalam tables.conf, kondisikan sesuai dengan storage partisi yang ada. Pada kasus ini, karena partisi lain terdapat hanya satu yaitu /dev/sdb dan dapat menampung hasil convert table secara keseluruhan maka tulis 40.

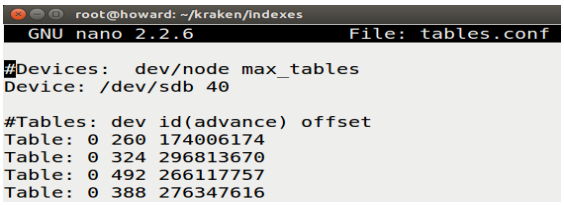


Gambar 4 - 12 Alokasi partisi rainbow table yang akan di convert

4.1.15 Convert Rainbow Table

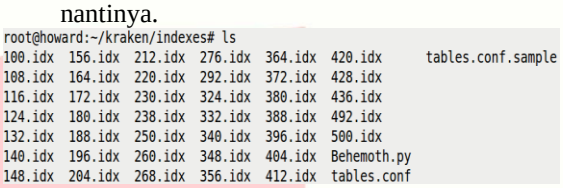
Berikut langkah-langkah melakukan convert rainbow table.

1. Melakukan *convert rainbow table* dengan menggunakan tool Behemoth.py. *Convert rainbow table* diperlukan karena kraken hanya dapat membaca file dalam bentuk raw dan juga tool Behemoth.py harus dijalankan dengan *mode root privileges*. Sebelum melakukan *convert rainbow table*, sangat perlu diperhatikan konfigurasi partisi yang akan menjadi tempat convertnya, partisi dimana *rainbow table* akan ditulis ulang harus kosong dan tanpa *filesystem*. Jalankan tool Behemoth.py dengan perintah berikut pada terminal ketik *sudo python Behemoth.py /home/howard/RT/* lalu *sudo python Behemoth.py /tempat disimpannya 40 file rainbow table format .dlt/*
2. Tool Behemoth.py akan membaca *rainbow table* format .dlt dari 1 sampai 40 dan ditulis ulang pada partisi tanpa *filesystem* yaitu /dev/sdb ketik *nano tables.conf*



Gambar 4 - 13 Alokasi partisi rainbow table setelah convert

3. Setelah proses convert table selesai, dalam direktori indexes akan ada file indexes. File indexes merupakan daftar isi dari kunci enkripsi yang nantinya akan digunakan pada kraken. File indexes digunakan untuk memudahkan kraken dalam pencarian Kc nantinya.



Gambar 4 - 14 File indexes yang muncul setelah convert

4. Masuk ke dalam direktori Kraken yang terdapat pada program kraken. Menghubungkan file indexes dengan rainbow table yang telah di convert lalu hubungkan *rainbow table* yang telah di convert pada /dev/sdb ke dalam file indexes dengan perintah berikut pada terminal ketik *cd kraken* lalu *./kraken ../indexes*

```
root@howard:~/kraken/Kraken# ./kraken ../indexes/
Device: /dev/sdb 40
/dev/sdb
Allocated 41301076 bytes: ../indexes//260.idx
Allocated 41259888 bytes: ../indexes//324.idx
Allocated 41239116 bytes: ../indexes//492.idx
Allocated 41257176 bytes: ../indexes//388.idx
Allocated 41269576 bytes: ../indexes//140.idx
Allocated 41237644 bytes: ../indexes//148.idx
Allocated 41404056 bytes: ../indexes//132.idx
```

Gambar 4 - 15 Menghubungkan file indexes dengan rainbow table

4.2 Pengujian

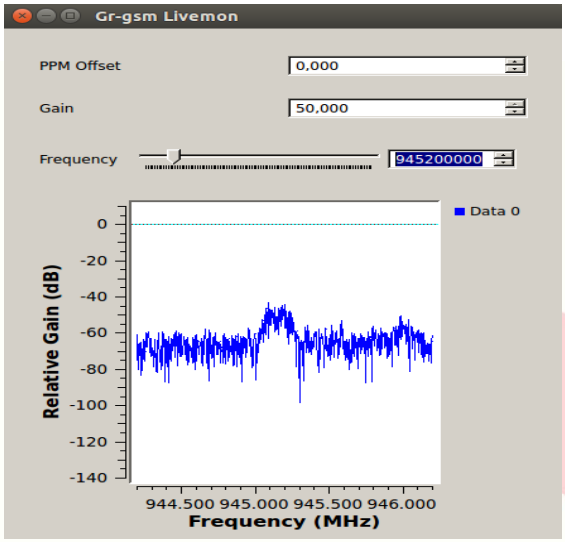
4.2.1 Pengujian Penangkapan Sinyal dan *Decoding* pada Operator Telkomsel dengan RTL-SDR, Gr-GSM dan Wireshark

Metode menggunakan perangkat RTL-SDR, aplikasi Gr-GSM, dan aplikasi wireshark yang mengikuti alur *decoding live* aplikasi Gr-GSM yang secara langsung menangkap sinyal dari BTS (*Base Transceiver Station*), menyimpan file yang berformat ".cfile", dan hingga decode data komunikasi GSM menggunakan Gr-GSM. Berikut langkah-langkah pengujiannya :

1. Menjalankan aplikasi Gr-GSM, aplikasi tersebut untuk menangkap sinyal GSM dari BTS, dengan mengetikkan perintah berikut pada terminal ketik *grgsm_livemon*
2. Menjalankan aplikasi wireshark karena aplikasi tersebut mendukung untuk analisis

paket data yang ditangkap oleh aplikasi Gr-GSM, dengan mengetikkan perintah berikut pada terminal ketik *wireshark*

3. Akan muncul kotak dialog grgsm yang dapat mengatur *gain* atau kekuatan penangkapan sinyal di "50,000" dan menetapkan frekuensi "945200000".



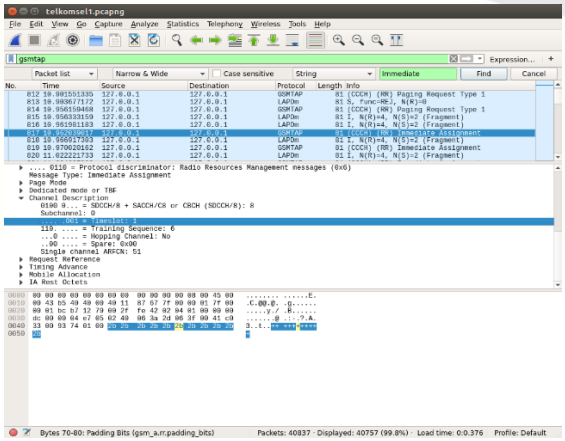
Gambar 4 - 16 Gr-GSM Livemon operator telkomsel

4. Pada tampilan terminal terlihat bahwa sinyal berhasil ditangkap. Outputannya berupa data bit yang akan dianalisis menggunakan wireshark.

```
root@howard:~# grgsm_livemon
gr-osmosdr v0.1.4-98-gc653754d (0.1.5git) gnuradio 3.7.12git-295-ga0adcd33
built-in source types: file osmosdr fcd rtl rtl_tcp uhd hackrf bladerf rfspace a
irspy soapy redpitaya
[INFO] [UHDlinux; GNU C++ version 4.8.4; Boost_105400; UHD_3.11.0.git-215-g3b206
caa]
Using device #0 Generic RTL2832U SN: 77771111153705700
Found Rafael Micro R820T tuner
[R82XX] PLL not locked!
Exact sample rate is: 2000000,052982 Hz
[R82XX] PLL not locked!
 15 06 21 00 01 f0 2b 2b 2b 2b 2b 2b 2b 2b 2b 2b 2b 2b 2b 2b 2b 2b
 d0 09 00 08 9c 7f 00 00 e0 6a f8 64 9c 7f 00 00 31 4c 80 d9 63 80 ff
 15 06 21 00 01 f0 2b 2b 2b 2b 2b 2b 2b 2b 2b 2b 2b 2b 2b 2b 2b 2b
 d0 09 00 08 9c 7f 00 00 e0 6a f8 64 9c 7f 00 00 31 4c 80 d9 63 80 ff
 d0 09 00 08 9c 7f 00 00 e0 6a f8 64 9c 7f 00 00 31 4c 80 d9 63 80 ff
```

Gambar 4 - 17 Gr-GSM Livemon pada terminal

5. Jika penangkapan berhasil dilakukan maka wireshark akan menampilkan data berupa paket yang isinya informasi dari frekuensi "945200000".



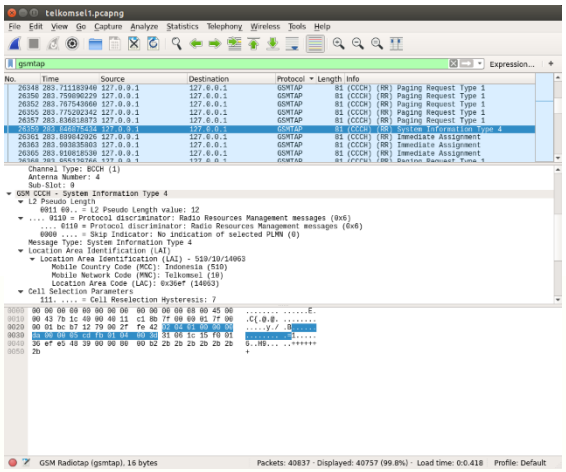
Gambar 4 - 18 Informasi frekuensi "945200000 pada wireshark

6. Lalu menyimpan data dalam format ".cfile" dapat menggunakan perintah "grgsm_capture" pada terminal, perintah "-f 945200000" untuk target frekuensi, perintah "-s 1000000" untuk tingkat sampel penangkapan yang sudah tetap dari program dengan variabel "1000000", perintah "-g 50" untuk kekuatan penangkapan sinyal, perintah "-c telkomsel1.cfile" data cfile yang akan disimpan, perintah "-T 60" untuk jarak waktu penangkapan sinyal, ketikkan *grgsm_capture –f 945200000 –g 50 –s 1000000 –c telkomsel1.cfile –T 60*
7. Apabila proses penyimpanan sudah selesai, maka data akan tersimpan otomatis di direktori /root/

```
root@howard:~# ls
airprobe                                    keystreamtelkomsel1
amdappsdk                                   kraken
calpp-0.90                                  pybombs
calpp-0.90.tar.gz                           pyrit
fglrx_15.201-0ubuntu1_amd64_UB_14.01.deb    rtl-sdr
fglrx-amdcccle_15.201-0ubuntu1_amd64_UB_14.01.deb  sms_multirtl_downlink_tail.cfile
fglrx-core_15.201-0ubuntu1_amd64_UB_14.01.deb      sms_multirtl_downlink_tail.txt
fglrx-dev_15.201-0ubuntu1_amd64_UB_14.01.deb       telkomsel1.cfile
gr-gsm                                      telkomsel1.pcapng
gsmframecoder                               telkomsel1.txt
kalibrate-rtl
```

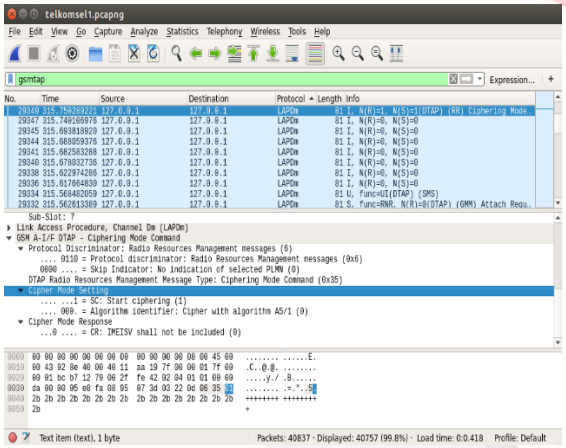
Gambar 4 - 19 Tampilan data telkomsel pada direktori

8. Menjalankan perintah "grgsm_decode" pada terminal untuk melakukan *decoding* pada data operator telkomsel yang sudah disimpan, disini melakukan *decoding* pada dua mode BCCH dan SDCCH8, BCCH yaitu mode saluran jaringan logikal yang digunakan BTS dalam GSM untuk mengirim informasi tentang identitas jaringan dimana informasi tersebut digunakan oleh *mobile station* untuk mendapatkan akses ke jaringan, kemudian SDCCH8 untuk mode jaringan yang melakukan sub aliran jaringan untuk pensinyalan. Kemudian "-a" adalah nomor unik yang diberikan untuk setiap frekuensi GSM tidak berbeda dengan frekuensi sebenarnya, "-s" adalah *samplerate* dimana batas frekuensi yang dapat dikirim perdetiknya, "-t 0" untuk *decoding* pada mode BCCH yang dimaksudkan adalah *timeslot* atau saluran lalulintas, ketikkan *grgsm_decode –a 51 –s 1000000 –c telkomsel1.cfile –t 0 –m BCCH* dan *grgsm_decode –a 51 –s 1000000 –c telkomsel1.cfile –t 1 –m SDCCH8*
9. Menjalankan aplikasi wireshark untuk membaca informasi yang didapatkan dalam bentuk paket informasi. Salah satu informasi yang didapatkan saat menjalankan mode decoding BCCH adalah identitas yang didapatkan dari data.



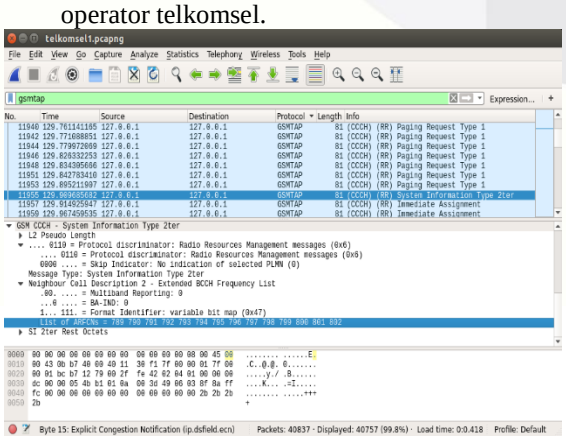
Gambar 4 - 20 Identitas operator jaringan telkomsel

10. Pada saat mendecoding mode SDCCH8 banyak informasi yang didapatkan, seperti mode enkripsi keamanan jaringan GSM yang digunakan operator telkomsel. Disini operator telkomsel menggunakan keamanan jaringan A5/1.



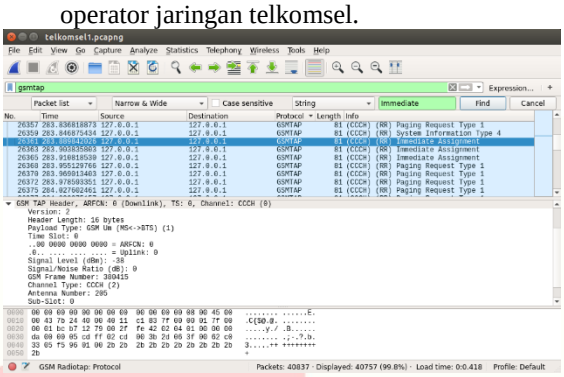
Gambar 4 - 21 Mode jaringan operator telkomsel

11. Pada decoding mode BCCH juga didapatkan informasi ARFCN (*Absolute Radio Frequency Channel Number*) frekuensi yang digunakan operator telkomsel.

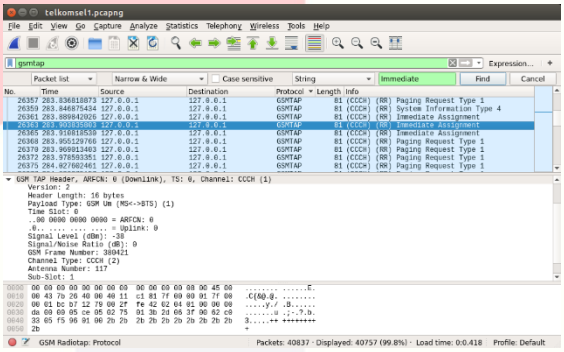


Gambar 4 - 22 ARFCN yang digunakan operator telkomsel

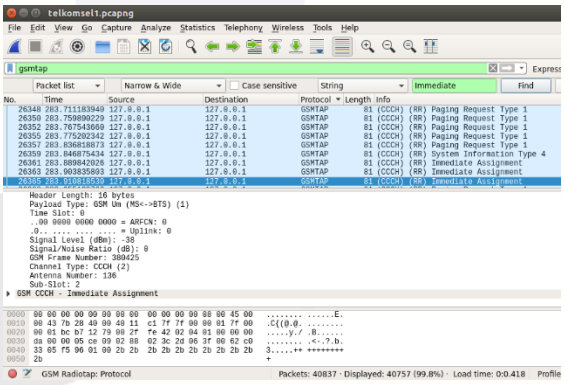
12. Dari mode decoding BCCH didapatkan juga informasi, GSM *Frame Number* yang berbeda karena penangkapan yang dilakukan secara *broadcast* atau pengiriman data ke banyak jaringan, berikut GSM Frame number pada operator jaringan telkomsel.



Gambar 4 - 23 GSM Frame Number 1 pada telkomsel



Gambar 4 - 24 GSM Frame Number 2 pada telkomsel



Gambar 4 - 25 GSM Frame Number 3 pada telkomsel

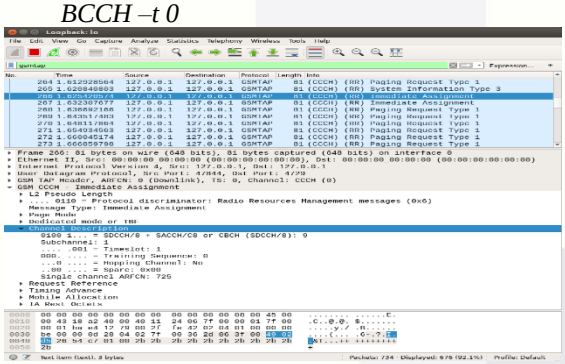
4.3.2 Pengujian *Decoding* pada *Sample Cfile* dengan RTL-SDR, Gr-GSM, dan Wireshark

Metode menggunakan perangkat RTL-SDR, aplikasi Gr-GSM, dan aplikasi wireshark yang mengikuti alur *decoding* aplikasi Gr-GSM yang tidak secara langsung menangkap sinyal dari BTS (*Base Transceiver Station*), file yang berformat ".cfile" telah diunduh dari website dan decode data komunikasi GSM menggunakan Gr-GSM. Berikut langkah-langkah pengujiannya :

1. Menjalankan aplikasi wireshark karena aplikasi tersebut mendukung untuk analisis paket data yang ditangkap oleh aplikasi Gr-

GSM, dengan mengetikkan perintah berikut pada terminal *wireshark.*

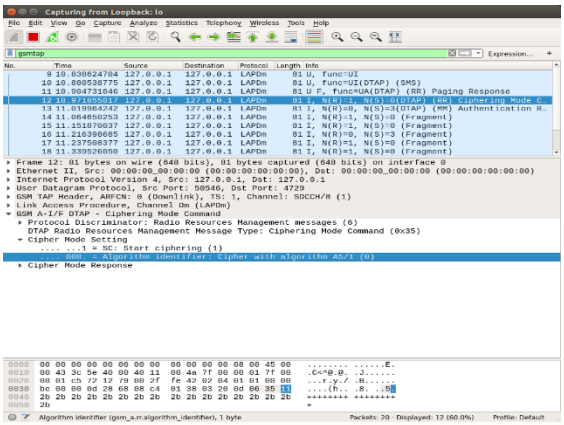
2. Menjalankan perintah "grgsm_decode" pada terminal untuk melakukan *decoding* pada *sample cfile* yang sudah disimpan, untuk melakukan *decoding* pada tiga mode BCCH, SDCCH8 dan TCHF. BCCH yaitu mode saluran jaringan logikal yang digunakan BTS dalam GSM untuk mengirim informasi tentang identitas jaringan dimana informasi tersebut digunakan oleh *mobile station* untuk mendapatkan akses ke jaringan, SDCCH8 untuk mode jaringan yang melakukan sub aliran jaringan untuk sinyal dan TCHF yaitu mode saluran tingkat penuh pada GSM yang diidetifikasikan sebagai saluran besar dengan kecepatan bit 22.8Kbps. Saluran ini memiliki dua arah yang memungkinkan untuk transfer suara atau pertukaran saluran. Kemudian pada *sample cfile* telah didapatkan *ARFCN* "-a 725" adalah nomor unik yang diberikan untuk setiap frekuensi GSM tidak berbeda dengan frekuensi sebenarnya, "-s $((100000000/174))" adalah *samplerate* dimana batas frekuensi yang dapat dikirim perdetiknya, "-t 0" untuk *decoding* pada mode "-m BCCH" yang dimaksudkan adalah *timeslot* atau saluran lalulintas, ketik *grgsm_decode –c vf_call6_a725_d174_g5_Kc1EF00BAB3BAC7002.cfile -s $((100000000/174)) –a 725 –m BCCH –t 0*



Gambar 4 - 26 Informasi dari *Sample cfile BCCH*

Pada gambar diatas informasi yang didapat dari *Immediate Assignment* pada *Channel Description* yaitu mode *decoding* SDCCH8 dan *timeslot 1*.

3. Kemudian melakukan *decoding* pada mode SDCCH8 dan *timeslot 1* yang telah didapatkan sebelumnya. *Restart* wireshark *capture* dan melakukan *decoding* SDCCH8. Hasil *decoding* pada wireshark, ketik *grgsm_decode –c vf_call6_a725_d174_g5_Kc1EF00BAB3BAC7002.cfile -s $((100000000/174)) –a 725 –m SDCCH8 –t 1*



Gambar 4 - 27 Informasi dari *Sample cfile* SDCCH8

Pada gambar diatas informasi yang didapatkan dari *Ciphering Mode Command* pada *Cipher Mode String* hanya enkripsi algoritma A5/1.

4.3.2 Pengujian Mencari *Bursts Frame* dari Data Komunikasi Operator Jaringan Telkomsel Menggunakan Airprobe

Burst Frame adalah sebuah data berupa sekumpulan bit yang berisikan informasi yang nantinya akan di dekripsi oleh aplikasi kraken, data yang didekripsi adalah KC (*key chipering*).

1. Masuk ke direktori "/airprobe/gsm-receiver/src/python" dan jalankan perintah "./go.sh" kemudian tambahkan perintah "/root/telkomsel1.cfile 64 1S" yang dimaksud sumber lokasi data telkomsel yang sudah ditangkap, 64 adalah tingkat tujuan GSM dan 1S adalah *timeslot* kemudian ditambah perintah "&>" adalah tujuan penyimpanan lokasi data burst, ketikkan *cd airprobe/gsm-receiver/src/python/* lalu *./go.sh /root/telkomsel1.cfile &> /root/telkomsel1.txt*
2. Setelah berhasil maka akan muncul pada direktori root.
3. Kemudian membuka isi data burst menggunakan perintah "nano telkomsel1.txt" pada terminal.

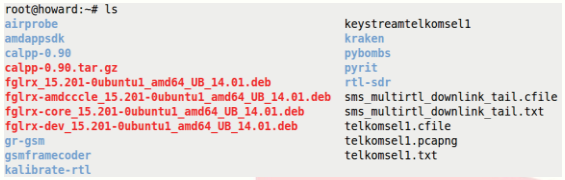
Gambar 4 - 28 Isi file burst data telkomsel

Data burst berhasil ditemukan, dan akan dilanjutkan ke proses dekripsi data oleh program kraken untuk mendapatkan KC (*key chipering*) pada data komunikasi GSM.

4.3.3 Pengujian Menggunakan GSM Framecoder pada *Bursts* Telkomsel

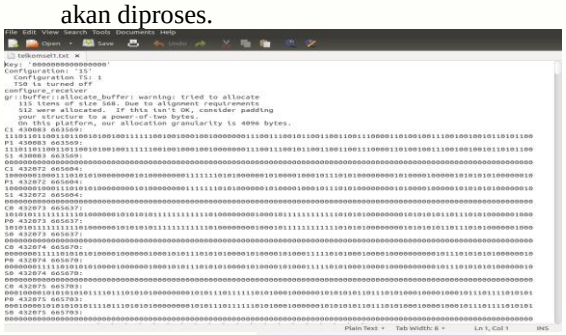
GSM Framecoder digunakan untuk menghitung *bursts* yang dihasilkan pada paket GSM untuk dapat di *decoding* datanya. Pada pengujian ini akan memproses file *bursts* operator Telkomsel yang telah didapatkan sebelumnya menggunakan aplikasi Airprobe.

1. Berikut tampilan file yang akan diproses pada direktori root. File tersebut bernama "telkomsel1.txt".



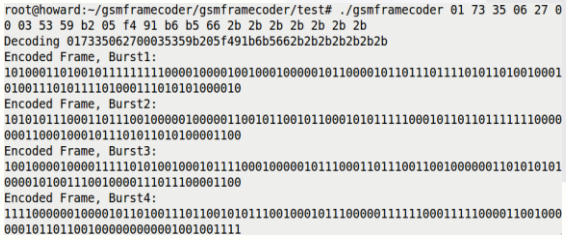
Gambar 4 - 29 Tampilan burst telkomsel pada direktori

2. Masuk ke direktori "gsmframecoder/gsmframecoder/test" menggunakan perintah berikut *cd gsmframecoder/gsmframecoder/test/*
3. Buka file burst dan melihat burst code yang akan diproses.



Gambar 4 - 30 Isi burst file telkomsel

4. Proses kode burst menggunakan perintah berikut.

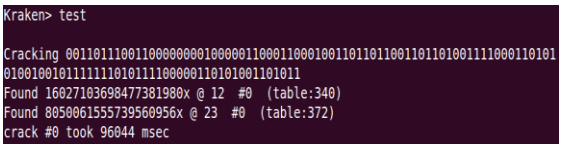


Gambar 4 - 31 Proses kode burst 1

Pada gambar diatas telah didapatkan *bursts frame* yang berbeda, dan akan diproses satu per satu menggunakan aplikasi kraken untuk mendapatkan KC (Kunci Chiper) dari file tersebut.

4.3.4 Ujicoba Rainbow Table

1. Untuk melakukan ujicoba kinerja rainbow table, ketik perintah berikut pada terminal *test*

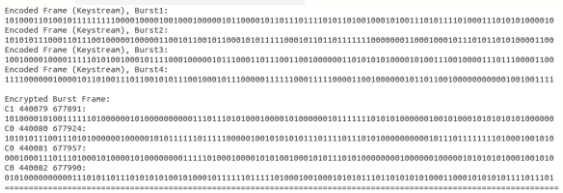


Gambar 4 - 32 Ujicoba rainbow table

2. Jika setelah melakukan test terdapat tabel yang ditemukan, maka rainbow table berjalan dengan baik.

4.3.5 Ujicoba xor keystream dengan Encrypted Burst

1. Masuk ke dalam direktori Utilities pada program kraken dengan ketikkan perintah berikut pada terminal *cd Utilities*
2. Pada percobaan kali ini akan menggunakan *tool* yaitu xor.py dan menggunakan *encrypted* burst dan *keystream* contoh. *Tool* ini berfungsi untuk menambahkan bit pada *keystream* dengan bit pada *encrypted* burst untuk mendapatkan XoR'ed burst, yang akan diberikan kepada kraken untuk di *crack*. Jalankan *tool* xor.py dengan ketikkan perintah berikut pada terminal *sudo python xor.py /masukkan keystream dan encrypted bursts/*
3. Terdapat *keystream* yang telah didapat dari gsmframecoder dan akan digabungkan dengan *encrypted* Burst ke dalam *tool* xor.py untuk mendapatkan *xor'ed burst*.



Gambar 4 - 33 Ujicoba xor encrypted burst pada telkomsel

Gambar 4 - 34 Ujicoba xor encrypted burst pada file contoh

4. Masukkan *keystream* yang telah didapat dan gabungkan dengan *encrypted* Burst ke dalam *tool* xor.py untuk mendapatkan *xor'ed Burst*.

Gambar 4 - 35 Ujicoba xor encrypted burst pada telkomsel

Gambar 4 - 36 Ujicoba xor encrypted burst pada file contoh

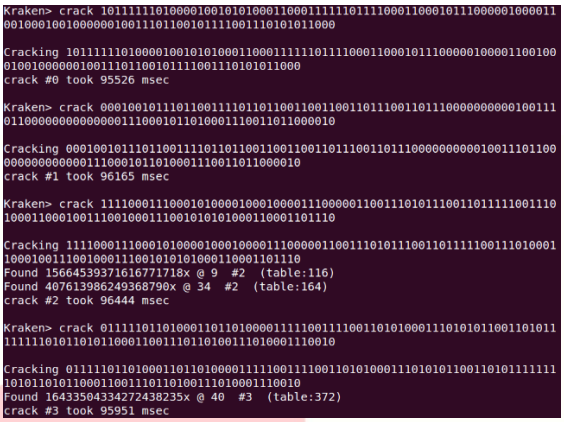
Pada gambar diatas telah didapatkan *XoR'ed bursts* yang berbeda, dan akan diproses satu per satu menggunakan aplikasi kraken untuk mendapatkan KC (Kunci Chiper) dari file tersebut.

4.3.6 Ujicoba crack xor'ed burst

1. Pengujian crack XoR'ed burst menggunakan data *encrypted* burst yang didapat dari *live decoding*. Pengujian *crack* XoR'ed burst berhasil di *crack*, apabila menampilkan output berupa *index* data yang digunakan untuk menunjukan kemungkinan kunci chiper dari burst tersebut. Untuk melakukan ujicoba dengan burst, ketik perintah berikut ini pada terminal *crack /masukkan xor'ed burst/*

Gambar 4 - 37 Ujicoba crack xor'ed burst telkomsel

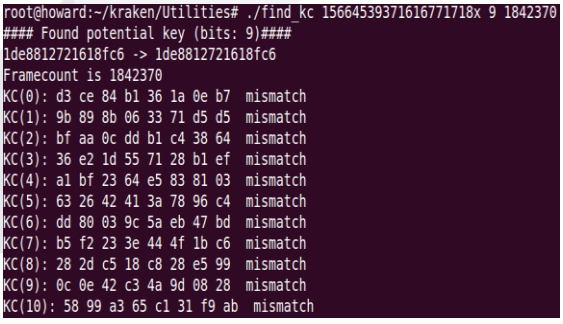
Pada gambar diatas *xor'ed bursts* dari telkomsel yang telah di *crack* tidak menampilkan index data yang digunakan untuk menunjukan kemungkinan kunci chiper dari burst tersebut.



Gambar 4 - 38 Ujicoba crack xor'ed burst pada file contoh

4.3.7 Ujicoba mencari kunci chiper dengan tool find_kc

1. Menjalankan *tool* find_kc dan masukkan data index dan *frame number* yang telah di *crack* untuk mendapatkan potensial Kc-nya, masuk ke direktori dimana *tool* find_kc berada yaitu direktori *Utilities* pada kraken. Untuk menjalankannya ketik perintah berikut pada terminal *cd Utilities*
2. Masukkan data index dan *frame number* burst yang telah didapat untuk mendapatkan potensial Kc-nya. Untuk menjalankannya ketik perintah berikut pada terminal *./find_kc /index data/frame number*



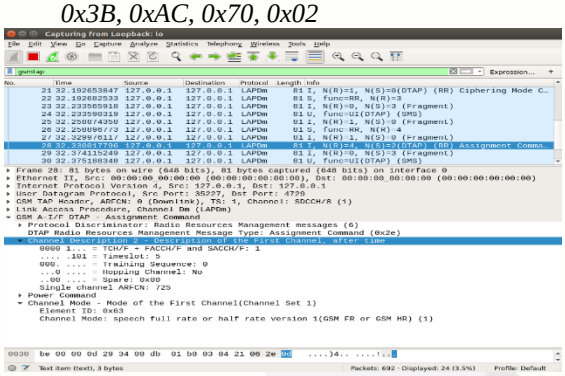
Gambar 4 - 39 Ujicoba mencari kunci chiper dari file contoh

3. Untuk mendapatkan kunci chiper dari xor'ed burst yang telah di *crack*, masukkan hasil data index yang telah dididapatkan dan masukkan *frame number* dan burst urutan sebelumnya untuk mengeliminasi Kc yang salah.

Gambar 4 - 40 Ujicoba mencari kunci chiper pada file contoh

4.3.8 Decoding Voice Call

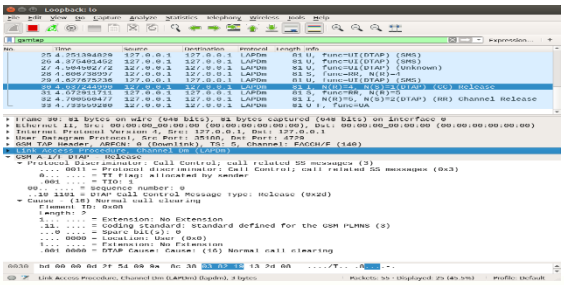
1. Dalam melakukan *decoding voice call* akan melakukan *decoding* SDCCH8 dengan menambahkan "-e 1" yaitu enkripsi algoritma A5/1 dan "-k 01E, 0xF0, 0x0B, 0xAB, 0x3B, 0xAC, 0x70, 0x02" yaitu kunci chiper yang didapatkan dari *mobile phone.* Wireshark akan menampilkan hasil *decoding.* Ketikkan perintah pada terminal *grgsm_decode –c vf_call6_a725_d174_g5_Kc1EF00BAB3BAC7 002.cfile -s $((100000000/174)) –a 725 –m SDCCH8 –t 1 –e 1 –k 01E, 0xF0, 0x0B, 0xAB, 0x3B, 0xAC, 0x70, 0x02*



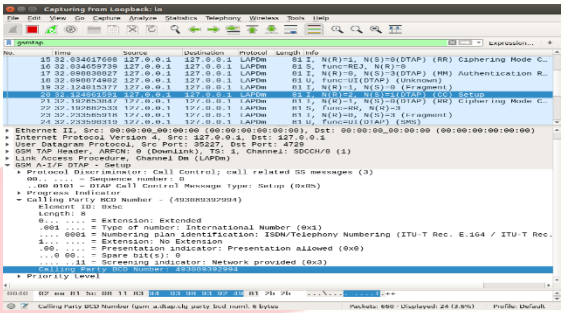
Gambar 4 - 41 Informasi dari *Sample cfile timeslot 5*

Pada gambar diatas terdapat informasi dari *Assignment Command* pada *Channel Description 2* yaitu pesan suara pada *timeslot 5* dan GSM-FR (*Voice Traffic Channel).*

2. Dalam melakukan *decoding voice call* akan menggunakan mode "-m TCHF" yaitu saluran sinyal dengan nilai penuh, diikuti dengan "-t 5" yaitu *timeslot* yang digunakan pada saluran suara dan "-d FR" yaitu kode suara yang didapatkan pada saluran, lalu "-o /tmp/speech.au.gsm" hasil *decoding voice traffic channel.* Ketikkan perintah di terminal *grgsm_decode –c vf_call6_a725_d174_g5_Kc1EF00BAB3BAC7 002.cfile -s $((100000000/174)) –a 725 –m TCHF –t 5 –e 1 –k 01E, 0xF0, 0x0B, 0xAB, 0x3B, 0xAC, 0x70, 0x02 –d FR –o /tmp/speech.au.gsm*

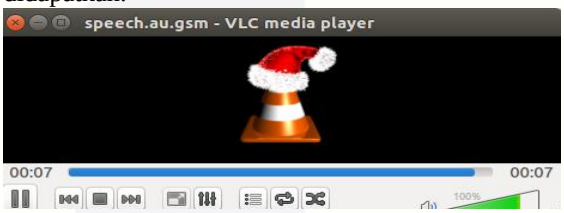


Gambar 4 - 42 Informasi dari *Sample cfile decoding voice call*



Gambar 4 - 43 Informasi dari *Sample cfile Calling Party (BCD) Number*

Setelah proses *decoding* selesai, hasil berada di direktor */tmp/speech.au.gsm* dan aplikasi VLC yang mendukung untuk membuka pesan suara yang didapatkan.



Gambar 4 - 44 Hasil decoding *sample cfile*

4.3.9 Hasil analisis

Dari hasil analisis ini dapat disimpulkan bahwa,

1. Implementasi *GPU Processing System* pada Ubuntu 14.04 LTS telah berhasil dilakukan dengan cara melakukan instalasi *graphic driver*, AMDAPPSDK, cal++, pyrit agar GPU yang digunakan dapat terdeteksi pada Ubuntu 14.04 LTS dan bekerja dengan baik.
2. Implementasi *GPU Processing System* pada dekripsi data komunikasi GSM telah berhasil dilakukan untuk mendapatkan kunci chiper dan membuka data komunikasi GSM yang berupa pesan suara. Dengan melakukan instalasi kraken, lalu melakukan *convert rainbow table* yang berisi kumpulan kunci chiper dan proses *convert rainbow table* dilakukan dalam waktu kurang dari 3 hari.

Pengujian pada proyek akhir ini tidak berhasil dilakukan dengan menggunakan *encrypted* burst dari *live decoding.* Akan tetapi, menggunakan alternatif yaitu *encrypted* burst yang terdapat pada : https://www.crazydanishhacker.com/gsm-cracking-

sms-kraken-software-defined-radio-series-16/ dan cfile yang terdapat di https://github.com/ptrkrysik/gr-gsm/wiki/Usage:-Decoding-How-To *Encrypted* burst contoh tersebut telah berhasil di xor dan didapatkan kunci chipernya dan *sample cfile* tersebut telah berhasil melakukan dekripsi data berupa *voice call* pada jaringan komunikasi GSM dengan menggunakan kunci chiper yang didapatkan dari *mobile phone* untuk menunjukkan bahwa enkripsi pada GSM terbukti benar memiliki kelemahan. Oleh karena itu, kunci chiper jenis ini sudah tidak aman lagi digunakan untuk meng-enkripsi data pengguna jaringan tersebut, dan disarankan kepada para pengguna jaringan GSM agar lebih memberi perhatian terhadap keamanan datanya dan mulai berpindah ke jaringan yang mempunyai enkripsi pada data yang lebih baik seperti UMTS (3G) LTE (4G).

5. Penutup

5.1 Kesimpulan

Dari hasil analisis ini dapat disimpulkan bahwa,

1. Implementasi GPU Processing System pada Ubuntu 14.04 LTS telah berhasil dilakukan dengan cara melakukan instalasi graphic driver, AMDAPPSDK, cal++, pyrit agar GPU yang digunakan dapat terdeteksi pada Ubuntu 14.04 LTS dan bekerja dengan baik.
2. Implementasi GPU Processing System pada dekripsi data komunikasi GSM telah berhasil dilakukan untuk mendapatkan kunci chiper dari file contoh dan membuka data komunikasi GSM yang berupa pesan suara dari *sample cfile*. Dengan melakukan instalasi git, dan kraken, lalu melakukan convert rainbow table yang berisi kumpulan kunci chiper dan proses convert rainbow table dilakukan dalam waktu kurang dari 3 hari.

5.2 Saran

Pada jaringan GSM, layanan yang ditawarkan pada jaringan ini sangat memuaskan pengguna jaringan tersebut. Akan tetapi, dari hasil analisis ini kunci chiper jenis ini sudah tidak cocok lagi digunakan untuk meng-enkripsi data pengguna jaringan tersebut. Oleh karena itu, disarankan kepada para pengguna jaringan GSM agar lebih memberi perhatian terhadap keamanan datanya dan mulai berpindah ke jaringan yang mempunyai enkripsi pada data yang lebih baik seperti UMTS (3G) LTE (4G).

Daftar Pustaka:

[1] S. M. Redl and M. K. Weber, "An Introduction to GSM," Artech House, March 1995.

[2] Ubuntu, "14.04," 2017. [Online]. Available: https://wiki.ubuntu.com/TrustyTahr/ReleaseNotes/14.04.5. [Accessed 27 03 2017].

[3] D. A. WINARTO, IMPLEMENTASI DAN ANALISIS ALGORITMA PARALEL FUZZY C-MEANS CLUSTERING DENGAN PENDEKATAN GRAPHICS PROCESSING UNITS (GPU), Bandung: Universitas Telkom, 2015, p. 10.

[4] R. McMillan, "New 'Kraken' GSM-cracking software is released," 21 07 2010. [Online]. Available: http://www.computerworld.com/article/2519495/mobile-wireless/new--kraken--gsm-cracking. [Accessed 08 03 2017].

[5] S. Mayer, "Rainbow Tables," *Breaking GSM with Rainbow Tables*, p. 2, 2010.

[6] Biryukov, Alex; Shamir, Adi; Wagner, David, "A5/1," *Real Time Cryptanalysis of A5/1 on a PC*, p. 1, 1978.

[7] D. Lestari and M. Z. Riyanto, "SUATU ALGORITMA KRIPTOGRAFI STREAM CIPHER BERDASARKAN FUNGSI CHAOS," *Algoritma Kriptografi Stream Cipher*, p. 34.

[8] Arpaci-Dusseau and R. H, "Hard Disk Drive," in *"Operating Systems; Three Easy Pieces, Chapter: Hard Disk Drives"*, Arpaci-Dusseau Books, 2014.

[9] L. Torvalds, "git," Kernel, [Online]. Available: https://git.kernel.org/pub/scm/git/git.git/tree/. [Accessed 24 07 2017].

[10] J. Stender, Aplikasi Platform Komputasi Software-Defined Radio (SDR) untuk Digital Spectrum Analizer, Malang, 2014.

[11] GNURadio, "PyBOMBS," Redmine, 2013. [Online]. Available: http://gnuradio.org/redmine/projects/pybombs/wiki. [Accessed 02 March 2017].

[12] RTL-SDR, "HOW TO CALIBRATE RTL-SDR USING KALIBRATE-RTL ON LINUX," RTL-SDR, 5 June 2013. [Online]. Available: http://www.rtl-sdr.com/how-to-calibrate-rtl-sdr-using-kalibrate-rtl-on-linux/. [Accessed 5 March 2017].

[13] R. Primartha, "Penerapan Enkripsi Dan Dekripsi File Menggunakan Algoritma Data Encryption (DES)," *Jurnal Sistem Informasi*, vol. III, p. Okrober, 2011.

[14] G. Radio, "Introduction to GNU Radio and Software Radio," gnuradio, 2006-2013. [Online]. Available: http://gnuradio.org/redmine/projects/gnuradio/wiki/Guided_Tutorial_Introduction. [Accessed 13 Maret 2017].

[15] M. F. R. D. R. S. Diki Nugraha, "IMPLEMENTASI GNURADIO GSM (gr-gsm) untuk decoding sinyal GSM".

[16] Wikipedia, "pip (package manager)," Wikipedia, 11 July 2017. [Online]. Available: https://en.wikipedia.org/wiki/Pip_(package_manager). [Accessed 16 July 2017].

[17] P. Krysik, "Gr-GSM," Github, 18 August 2015. [Online]. Available: https://github.com/ptrkrysik/gr-gsm/wiki. [Accessed 19 July 2017].

[18] RTL-SDR, "SNIFFING AND ANALYZING GSM SIGNALS WITH GR-GSM," RTL-SDR, 1 December 2014. [Online]. Available: http://www.rtl-sdr.com/tag/airprobe/. [Accessed 23 July 2017].

[19] K. Meier, "[A51] gsmframecoder to calculate bursts," Srlabs.de, 23 February 2011. [Online]. Available: https://lists.srlabs.de/pipermail/a51/2011-February/001067.html. [Accessed 23 July 2017].

[20] A. Parmar and K. M. Pattani, "Sniffing GSM Traffic Using RTL-SDR And Kali Linux OS," *Sniffing GSM Traffic Using RTL-SDR And Kali Linux OS*, vol. IV, no. 01, pp. 1637-1642, 2017.