SIMULASI SISTEM *SCRAMBLING* VIDEO DENGAN TEKNIK PENGACAKAN *PIXEL* DAN *ADDRESS CODE*

SKRIPSI

Oleh :

Arif Astomo

04 04 03 0148

DEPARTEMEN ELEKTRO

FAKULTAS TEKNIK UNIVERSITAS INDONESIA

GENAP 2007/2008

SIMULASI SISTEM *SCRAMBLING* VIDEO DENGAN TEKNIK PENGACAKAN *PIXEL* DAN *ADDRESS CODE*

SKRIPSI

Oleh :

Arif Astomo

04 04 03 0148

SKRIPSI INI DIAJUKAN UNTUK MELENGKAPI SEBAGIAN PERSYARATAN MENJADI SARJANA TEKNIK

DEPARTEMEN ELEKTRO

FAKULTAS TEKNIK UNIVERSITAS INDONESIA

GENAP 2007/2008

PERNYATAAN KEASLIAN SKRIPSI

Saya menyatakan dengan sesungguhnya bahwa skripsi dengan judul :

SIMULASI SISTEM *SCRAMBLING* VIDEO DENGAN TEKNIK PENGACAKAN *PIXEL* DAN *ADDRESS CODE*

Yang dibuat untuk melengkapi sebagian persyaratan menjadi sarjana Teknik pada Program Studi Teknik Elektro Departemen Teknik Elektro Fakultas Teknik Universitas Indonesia, sejauh yang saya ketahui bukan merupakan tiruan atau duplikasi dari skripsi yang sudah dipublikasikan dan atau pernah dipakai untuk mendapatkan gelar kesarjanaan di lingkungan Universitas Indonesia maupun di Perguruan Tinggi atau Instansi manapun, kecuali bagian yang sumber informasinya dicantumkan sebagaimana mestinya.

Depok, Juni 2008

Arif Astomo

NPM 04 04 03 0148

PENGESAHAN

Skripsi dengan judul:

SIMULASI SISTEM *SCRAMBLING* VIDEO DENGAN TEKNIK PENGACAKAN *PIXEL* DAN *ADDRESS CODE*

Dibuat untuk melengkapi sebagian persyaratan menjadi Sarjana Teknik pada program studi Teknik Elektro Departemen Teknik Elektro Fakultas Teknik Universitas Indonesia dan disetujui untuk diajukan dalam sidang ujian skripsi.

Depok, Juni 2008

Dosen Pembimbing,

Fitri Yuli Zulkifli, ST, M.Sc.

NIP. 131 476 472

UCAPAN TERIMA KASIH

Segala puji bagi Allah SWT atas segala limpahan rahmat dan kasih sayang-Nya sehingga penulisan skripsi ini dapat terselesaikan dengan baik. Shalawat serta salam semoga selalu tercurah kepada Baginda Rasul Muhammad SAW beserta para pengikutnya. Penulis juga mengucapkan banyak terima kasih kepada :

1. Ibu Fitri Yuli Zulkifli, ST, M.Sc dan Bapak Arman Djohan Diponegoro, M.Eng selaku dosen pembimbing yang telah mencurahkan banyak waktunya untuk memberikan pengarahan, masukan, pengkoreksian, kritikan yang membangun, serta bimbingan selama masa penulisan hingga terselesaikannya penulisan skripsi ini.
2. Kedua orang tua penulis, Bapak Aris Santoso dan Ibu Sri Astuti Arsih, yang terus memberikan dukungan dan doanya hingga terselesaikannya skripsi ini.
3. Kakak penulis, Arsanti Tyastuti, yang telah memberikan berbagai dukungan baik fisik maupun moril selama penulisan ini.
4. Rekan-rekan Elektro 2004 Anggi Purwanto, Yani Barliani, Ulfa Dwi U, Iyung, Apul Luthfi, Mahadhir, Dicky Jonathan, Aji Teguh P, Afita Putri L, Dwi Putri P dan rekan-rekan lainnya yang namanya tidak bisa penulis sebutkan satu persatu.
5. Asisten Lab Kendali Dayat dan Ayok.
6. Teman terdekat penulis, Ayu Diah Lestari, yang selalu memberikan dukungan dan doanya serta kesetiaan menemani penulis baik siang maupun malam.

Depok, Juni 2008

Penulis

Arif Astomo
NPM 04 04 03 0148
M.Sc.
Departemen Teknik Elektro

Dosen Pembimbing
Fitri Yuli Zulkifli, ST,

SIMULASI SISTEM *SCRAMBLING* VIDEO DENGAN TEKNIK PENGACAKAN *PIXEL* DAN *ADDRESS CODE*

ABSTRAK

Keinginan dari para pemegang hak cipta dalam penyiaran program-program *premium*, telah memicu perusahaan TV kabel untuk membuat suatu sistem keamanan dalam penyiaran program tersebut. Salah satu sistem keamanan yang dibuat adalah dengan memanipulasi sinyal video yang akan disiarkan atau yang dikenal dengan istilah *scrambling* video.

Saat ini telah terdapat berbagai macam metode *scrambling* video, diantaranya adalah *traps, video inversion, interfering carrier, horizontal sync suppression* dan *digital audio encryption*. Akan tetapi, metode-metode tersebut sistem keamanannya hanya memanfaatkan kelemahan proses *detuning* dari TV. Sehingga dengan menghilangkan kelemahan tersebut, para pembajak video dapat dengan mudah memperoleh siaran program-program *premium*.

Pada skripsi ini, akan dirancang suatu simulasi sistem *scrambling* video dengan menggunakan teknik pengacakan pixel yang digabungkan dengan suatu *address code* melalui sistem operasi matriks. Selanjutnya hasil *scrambling* video akan divariasikan dengan besar *sample time* mulai dari 1/15 hingga 1/60, serta jumlah *delay* dari tiap *sample frame* video yang disiarkan. Proses analisis akan dilihat pada tampilan *video display* dan *vector scope*.

Hasil analisis simulasi menunjukkan bahwa tingginya tingkat keamanan dari *scrambling* video ditentukan oleh banyaknya pixel video yang terbagi, penggunaan manipulator-manipulator matriks dan acaknya nilai pembangkitan matriks oleh *address code.* Variasi *sample time* 1/15 akan memberikan hasil distorsi yang terbesar, sedangkan *sample time* 1/60 akan memberikan pengurangan kedipan video dan hasil distorsi yang terkecil.

Kata kunci : *scrambling* video, pengacakan pixel, *address code*, *sample time*

Arif Astomo
NPM 04 04 03 0148
Departemen Teknik Elektro

Dosen Pembimbing
Fitri Yuli Zulkifli, ST, M.Sc.

SIMULATION VIDEO *SCRAMBLING* SYSTEM WITH RANDOM *PIXEL* TECHNIQUE AND *ADDRESS CODE*

ABSTRACT

The demand from the rights holder in broadcasting their premium programs has triggered TV cable company to make a security system for those programs. One of the security system that has been made is by manipulating the video signals or been known by the word of video *scrambling*.

Until now, there are so many method of video *scrambling*, like *traps, video inversion, interfering carrier, horizontal sync suppression* dan *digital audio encryption*. However, those security system methods are only using the weaknesses of *detuning* process by the television set. So that, by eliminating these weaknesses, the hijacker can get those premium programs easily.

In this semi thesis, will be designed a simulation video *scrambling* system with random pixel technique that has been combined with a unique *address* code based on matrix operation system. Furthermore, the result of *scrambling* video will be varied with adjusting the *sample time from* 1/15 to 1/60 and varying the number of *delay* of each *sample* video *frame* that will be broadcasted. The analysis process will be showed in the *video display* and *vector scope.*

The simulation analysis result shows that security of *scrambling* video is depends on the number of video pixel divided, the use of matrix manipulators and randomize the value of matrix generator by *address code*. The *sample time* 1/15 will gives the largest distortion, whereas *sample time* 1/60 will gives reduction in video *flicker* and smallest distortion.

Keywords : *scrambling* video, random pixel, *address code*, *sample time*

DAFTAR ISI

DAFTAR GAMBAR

DAFTAR TABEL

DAFTAR ISTILAH

Bit	Bilangan sistem biner yang terdiri atas angka 0 dan 1 (satu). Dalam sistem telekomunikasi, bit digunakan untuk merepresentasikan informasi dalam bentuk kode
Integer	Suatu bilangan logika yang terdiri dari angka 0 hingga 255. Tipe bilangan ini biasa dipakai untuk merepresentasikan jenis data video ataupun untuk konversi kode dari bit ke integer dan sebaliknya.
Frame per sekon	Banyaknya jumlah *frame* atau bingkai gambar yang ditampilkan pada setiap urutan perwaktuan. Makin tinggi *fps*, maka makin bagus pula gambar yang terbentuk (sedikit kedipan).

DAFTAR SINGKATAN

RGB	*Red, Green, Blue*
I	*Intensity*
SYNC	*Synchronizing*
A/D	*Analog to Digital*
D/A	*Digital to Analog*
PROM	*Programmable Read Only Memory*
EEPROM	*Electrical Eraseable PROM*
FSK	*Frequency Shift Keying*
PSK	*Phase Shift Keying*
OSC	*Oscillator*
AC	*Alternating Current*
DC	*Direct Current*
Hz	*Hertz*
HBI	*Horizontal Blanking Interval*
VBI	*Vertical Blanking Interval*
OOB	*Out-Of-Band*
dB	*Decibel*
AGC	*Automatic Gain Control*
ATTN	*Attenuation*
VSB	*Vestigal Side Band*
S/N	*Serial Number*
CH	*Channel*
CPLD	*Complex Programmable Logic Device*
CATV	*Cable TV*
BPF	*Band Pass Filter*
IF	*Intermediate Frequency*
RF	*Radio Frequency*
BIT	*Binary Digit*
FIFO	*First-In First-Out*
Kbps	*Kilo Byte per Seconds*

μs	*Mikroseconds ($10^{-6}s$)*
MUX	*Multiplexer*
DEMUX	*Demultiplexer*
AM	*Amplitude Modulation*
FM	*Frequency Modulation*
Fl	*Frequency Low*
Fh	*Frequency High*

BAB 1 PENDAHULUAN

1.1. LATAR BELAKANG

Telekomunikasi merupakan hal yang mutlak dibutuhkan oleh semua orang. Pada awal mulanya, kebutuhan akan telekomunikasi dilakukan oleh masing-masing individu terhadap individu lainnya. Kemudian, telekomunikasi berkembang dengan menggunakan kawat/kabel. Hal ini dilakukan semata-mata untuk fleksibilitas dalam berkomunikasi. Kebutuhan akan telekomunikasi terus meningkat dari waktu ke waktu. Definisi dari telekomunikasi bukan hanya sekedar penyampaian suara/audio saja, tapi juga penyampaian informasi dalam bentuk gambar dan suara (audio/visual) [1]. Penyampaian informasi dalam bentuk audio/video ditayangkan sejak manusia menciptakan TV untuk pertama kali. Akan tetapi, penerimaan sinyal TV pada waktu itu sangatlah buruk. Hal ini membuat salah satu perusahaan, yang pada nantinya berkembang menjadi perusahaan TV kabel, menambahkan komponen tambahan agar pemancaran maupun penerimaan sinyal TV menjadi lebih baik [2].

Perusahaan TV kabel memasang tarif serta membatasi siaran mereka kepada kalangan/pemirsa yang ingin menikmati layanan mereka. Oleh karenanya, perusahaan TV kabel membuat agar proses pentransmisian maupun penerimaan data mereka dapat berlangsung aman tanpa adanya proses pembajakan. Sehingga dibuatlah proses manipulasi/*scrambling video* pada sisi pentransmisian dan proses kebalikannya/*descrambling video* pada sisi penerima.

Hingga saat ini, semua teknik scrambling video memanfaatkan kelemahan proses *detuning* dari TV [3]. Oleh karenanya, skripsi ini akan membahas tentang simulasi rancangan sistem *scrambling* video dengan menggunakan *address code* sebagai *unlocking key* dan teknik pengacakan pixel dari video yang disiarkan.

1.2. TUJUAN PENULISAN

Tujuan dari penulisan skripsi ini adalah memodelkan dan mensimulasikan proses *scrambling* video berdasarkan manipulasi sinyal video dengan teknik pengacakan pixel video dan penggunaan *address code* serta variasi yang dilakukan terhadap *sample time* dan *delay.*

1.3. BATASAN MASALAH

Batasan masalah dari penulisan skripsi ini adalah pembahasan tidak dilakukan pada tingkat pentransmisian video. Akan tetapi, dilakukan pada analisis *scramble* video dengan melihat hasil tampilan video dan bentuk grafik pada *vector scope* serta *frame rate display.* Tingkat uji coba simulasi proses *scrambling* video mulai dari pihak pengirim hingga ke pihak penerima dilakukan dengan penghubung biasa tanpa ada atenuasi/pelemahan sinyal. Komponen warna yang di*scramble* hanya bagian R-saja. *Sampling time default* yang digunakan adalah 10 s dan untuk 1 jenis video, 1 jenis pelanggan dan 1 konfigurasi *address code*.

1.4. METODOLOGI PENELITIAN

Metode penulisan yang digunakan dalam penyusunan skripsi ini adalah :

a. Tinjauan pustaka, yaitu dengan melakukan studi literatur dari buku-buku pustaka ataupun masalah yang terkait.
b. Diskusi, dengan melakukan pembahasan dengan pembimbing maupun pihak yang terkait akan skripsi tersebut.
c. Pengambilan bahan dari internet sebagai referensi.
d. Melakukan simulasi terhadap rancangan yang dibuat dan melakukan analisis terhadap rancangan tersebut.

1.5. SISTEMATIKA PENULISAN

Sistematika penulisan skripsi ini dibagi menjadi beberapa bagian, yaitu :

Bab I Pendahuluan

Bagian ini terdiri dari latar belakang masalah, tujuan penulisan, batasan masalah, metodologi penelitian dan sistematika penulisan

Bab II Prinsip *scrambling* video dengan sistem operasi Matriks

Bagian ini terdiri dari penjabaran dari masing-masing teknik *scrambling* yang sudah ada dan prinsip perhitungan matriks baik sebagai vektor maupun skalar

Bab III Konfigurasi video *scrambler*

Bagian ini terdiri dari konfigurasi hasil rancangan yang telah dibangun dengan memperlihatkan blok diagram dari sistem maupun dari sub-sistem.

Bab IV Analisis dan Uji Coba

Bagian ini terdiri dari uji coba hasil rancangan sekaligus melakukan analisis terhadap hasil *scramble* pixel, kontras gambar antara pihak pengirim dan pihak penerima, dan spektrum gambar.

Bab V Kesimpulan

BAB 2

PRINSIP *SCRAMBLING* VIDEO DENGAN SISTEM OPERASI MATRIKS

2.1. PRINSIP SCRAMBLING VIDEO

2.1.1. KONSEP UMUM

Sinyal televisi terdiri dari komponen audio dan video. Pentransmisian dapat di-*scramble* atau mengubah salah satu atau kedua komponen tersebut sehingga *receiver*/penerima televisi tidak dapat mengenali formatnya dan tidak dapat mereproduksi material aslinya. Teknik *scrambling* video harus memperhatikan beberapa hal seperti :

1. SECURITY LEVEL

Tingkat keamanan adalah tingkat kesulitan untuk orang yang tidak dikenal/*unauthorized person* yang mencoba untuk me-*recovery* sinyal asli televisi. Sistem yang paling aman umumnya diterapkan pada teknologi militer untuk menjaga kerahasiaan negara.

Meskipun sinyal audio dan video di-*scramble*, dalam kebanyakan sistem enkripsi, perlindungan melawan pembajakan biasanya dimulai dari tingkat keamanan dari 1 komponen dari sinyal komposit video. Saat ini, semua produk untuk penerima satelit televisi, sudah mendapatkan tingkat keamanan enkripsi sinyal informasi audio. Oleh karenanya, sistem kabel televisi memulai tingkat keamanan dengan metode untuk *scrambling* video. Sedangkan audio biasanya ditransmisikan dengan jernih tanpa di-*scramble*.

2. SCRAMBLING VIDEO

Scrambling informasi video adalah sebuah proses pengubahan karakter dari sinyal asli video tersebut. Informasi video tidak akan terpengaruh jika porsi non-*picture* dari sinyal seperti pulsa sinkronisasi dihilangkan, dihambat atau digeser dari posisi normalnya. Sedangkan informasi video dapat terpengaruh jika gelombang sinyalnya di-*invert*, pergeseran level tegangan pergeseran waktu (*time*

shifting), atau menambahkan sebuah sinyal interferensi untuk menutupi (*mask*) keseluruhan sinyal video televisi [4].

3. SCRAMBLING AUDIO

Metode yang aman untuk men-*scramble* sinyal audio saat ini sudah banyak tersedia. Hal ini disebabkan *bandwidth* untuk audio relatif lebih sempit / terbatas dan tingkat pentransmisian informasinya lebih rendah dari porsi untuk video.

Pada awalnya, peralatan untuk *scrambling* audio menggunakan teknik yang sederhana seperti memasukkan sebuah sub-carrier kedua (*secondary sub-carrier*) untuk menyembunyikan informasi audio. Atau me-remodulasi sub-carrier dasar audio ke frekuensi yang lebih tinggi.

Saat ini, dengan biaya yang rendah dan kecepatan yang lebih tinggi, sebuah A/D *converter* tersedia untuk men-transformasikan inpul analog audio menjadi aliran data digital yang dapat di-enkripsi ke level keamanan yang lebih tinggi. Data audio digital tersebut lalu digabungkan dengan control dan alamat informasi dan di "*embedded*" pada sinyal video untuk keperluan transmisi.

4. CONTROL and SUBSCRIBER AUTHORIZATION

Kontrol informasi tidak punya hubungan secara langsung pada pentransmisian data. Pertanyaan utama dalam mengevaluasi dan memahami sebuah sistem enkripsi adalah fungsi dari aliran data ini. Kontrol informasi dapat mengirim secara *kontinu* komponen yang meng-*enable* sebuah *descrambler* atau kontrol informasi secara sederhana digunakan untuk memberi hak/kuasa untuk para pelanggan untuk mendapatkan sinyal informasi.

Kontrol informasi dapat menentukan apakah sebuah decoder dapat menerima satu atau beberapa kombinasi channel dari privat televisi yang tersedia. Hal ini dikenal sebagai *tiers* [10]. Sangat jelas bahwasanya keseluruhan disain sistem dan interaksi antara data kontrol dan juga metode scrambling, akan menentukan keseluruhan tingkat keamanan yang akan dipakai.

5. ENCRYPTION and KEY DISTRIBUTION

Secara konsep, meng-enkripsi aliran data informasi adalah sederhana. Sebuah input diproses berdasarkan pada suatu formula, yang dikenal sebagai algoritma, lalu spesifikasinya ditentukan oleh sebuah "*kunci*". Baik kunci dan informasi yang telah dienkripsi ditransmisikan ke decoder yang menggunakan kunci dan algoritma yang sama untuk membuka data tersebut [11]. Hasil akhirnya adalah sebuah output yang sama dengan inputnya.

Contoh sederhana dari sebuah instruksi algoritma adalah jika semua data data telah diubah kedalam bentuk digital dan diekspresikan sebagai grup dari 4 bit, sebuah pilihan dari kemungkinan algoritma adalah meng-*ADD* sebuah kunci pada masing-masing grup. Jika kunci tersebut misalnya adalah 0001, dan data aslinya adalah :

0111 1110 0101 1010 1110

Data tersebut akan ditransformasikan menjadi penjumlahan kode biner. Sehingga menjadi :

1000 1111 0110 1011 1111

Kunci tersebut, 0001, lalu dikurangkan pada decoder agar dapat mereproduksi data aslinya. Karena kunci merupakan bentuk digital yang terdiri dari banyak bit, berbagai kemungkinan kunci pun tersedia. 2-bit dapat diatur menjadi 4 kemungkinan (2^2), 8-bit kunci dapat diatur menjadi 256 kemungkinan (2^8), dan 16-bit dapat diatur menjadi 65.536 (2^{16}) [13].

Berbagai macam cara telah dikembangkan tentang peng-enkripsian data. Karena keamanan pentransmisian informasi telah menjadi sangat penting untuk operasi militer dan mata-mata, sebuah metode encoding/decoding dikembangkan secara rahasia. Sebuah algoritma yang bagus adalah tahan/*resistant* terhadap kemungkinan kodenya untuk dipecahkan. Tidak mudah terdeteksi dan tidak jelas hubungannya antara input, output dan kunci yang digunakan. Meskipun pembajak mengetahui algoritma yang digunakan tapi tidak mempunyai akses terhadap kunci, akan membuat proses pembukaan data menjadi sangat sulit. Oleh karena itu, elemen utama atau keamanan itu sendiri sebenarnya terletak pada distribusi kunci.

Sistem yang diciptakan untuk distribusi kunci sama pentingnya seperti algoritma dari enkripsi. Tingkat keamanannya bergantung pada jumlah bit yang menyusun kunci tersebut, channel yang digunakan untuk mendistribusikan kunci dan siapa saja yang boleh/tidak boleh mengakses informasi video tersebut. Pada sistem yang lebih modern lagi, kunci itu sendiri juga dienkripsi dengan level yang lebih tinggi. Selain itu, kunci juga diganti secara periodik baik melalui decoder maupun melalui jaringan kontrol komputer.

Perlu disadari bahwasanya men-*scrambling* kunci ke level enkripsi yang lebih tinggi akan menciptakan kunci tambahan. Hal ini akan menciptakan level kesulitan yang lain untuk para pembajak.

6. ADDRESSING and TIERING

Dalam rangka untuk mencapai tingkat keamanan yang tinggi, sistem enkripsi menjadikan decoder sehingga mempunyai *address*/alamat tertentu. Hal ini sama seperti nomor pada telepon. Para pelanggan dapat diaktifkan/dinon-aktifkan dari kontrol pusat. Atau, dengan mengirimkan pesan layanan sesuai dengan status tagihan mereka.

Database pelanggan disimpan di dalam komputer pusat. Setiap pelanggan mempunyai sebuah nomor serial sendiri dan daftar layanan/channel yang telah dipesan dan dibayar. Sistem pengalamatan secara penuh (*fully* addressable) mengijinkan daftar layanan ini diubah sewaktu-waktu sesuai dengan keinginan pelanggan. Informasi ini disampaikan pada masing-masing decoder melalui data yang terkandung dalam sinyal TV. Karena channel pelanggan dapat diatur/diubah ke channel tertentu dan sinyal informasi harus ditransmisikan ke semua *encoding* channel, maka komputer pusat harus secara kontinu meng-*update* database dari semua pelanggan. Lalu, setiap decoder akan mendeteksi semua sinyal informasi yang ditransmisikan. Tapi akan merespon pada sinyal informasi yang nomor serinya bersesuaian dengan decoder pelanggan.

Kelemahan dari sistem scrambling pengalamatan penuh terletak pada ketidakmampuannya untuk membatasi aliran sinyal informasi. Setiap pelanggan menerima semua informasi yang ditransmisikan [4]. Jika terjadi error, kesalahan dalam memberikan kunci atau juga ”sesorang” menduplikasi alamat pelanggan

dari kabel TV, maka informasi yang sama akan dapat terakses ke semua decoder dengan nomor seri tersebut.

Alamat biasanya disimpan di dalam *PROM* atau *EEPROM*. EEPROM dapat menghapus memorinya dengan arus listrik. Lalu menulis ulang informasi yang baru ke memori tersebut. Oleh karena itu, decoder dirancang agar papan sirkuitnya tidak bisa di-print dan juga memorinya tidak dapat dengan mudah dihapus atau diduplikat.

7. OUT- OF- BAND ADDRESSING

Berbagai macam cara telah diciptakan untuk mentransmisikan alamat dan data lain dari computer pusat ke tiap decoder pelanggan. Salah satu caranya adalah dengan sub- carrier *out of band.* Pada metode tipe ini, data dimodulasi ke sebuah frekuensi carrier yang ditumpangkan di suatu area dalam ruang kosong (*blank space*) antara channel 4 dan 5 VHF [13]. Atau pada batas frekuensi yang lebih tinggi dari FM radio, yaitu pada 108 MHz. Lalu tiap decoder diatur ke frekuensi tengah carrier, sehingga aliran data dapat termonitor secara kontinu. Keuntungan dari sistem ini adalah transmisi data terpisah dari sinyal informasi yang akan dilihat oleh pelanggan. Kerugiannya adalah dibutuhkannya sub- carrier yang terpisah untuk pemancaran data. Sehingga harus mengalokasikan ekstra frekuensi. Selain itu, diperlukan juga penerima (*receiver*) kedua untuk menerima informasi *out- of- band.*

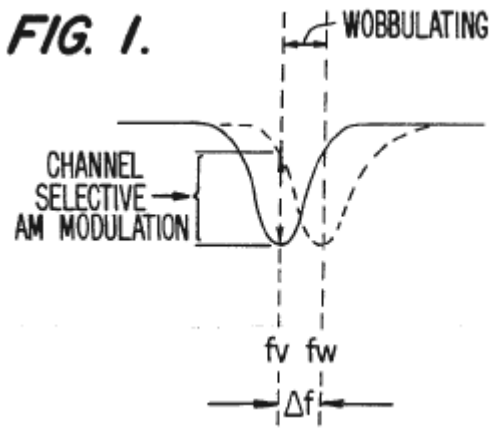
2.1.2. METODE SCRAMBLING

Metode enkripsi telah berkembang pesat hingga saat ini. Dimulai dari cara yang sederhana, seperti perangkap (*traps*), sampai cara yang paling kompleks yang melibatkan decoder dengan microchip komputer. Berikut adalah penjabaran dari masing-masing metode tersebut.

1. PERANGKAP (*TRAPS*)

Sesuai dengan namanya, peralatan ini adalah sebuah filter yang menangkap channel atau frekuensi dengan batas tertentu sementara melewatkan frekuensi yang lain. Ada 2 jenis *traps*, yaitu *traps* positif dan *traps* negatif. Trap negatif terdiri dari sekumpulan jaringan komponen pasif, seperti kapasitor dan induktor. Pada trap jenis ini, filternya memindahkan channel yang telah dipilih yang disuplai ke rumah pelanggan. Pada sistem perangkap negatif, perangkapnya dipusatkan pada carrier video yang tidak pernah mencapai pelanggan. Kebanyakan perangkap jenis ini mempunyai frekuensi yang sempit dan tidak menjangkau *bandwidth* TV, 6 MHz [5]. Trap negatif biasanya ditempatkan diluar rumah pelanggan pada kotak diatas tiang dekat tap kabel.

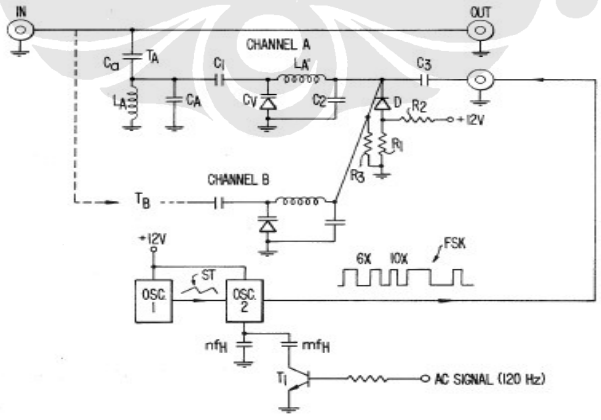
Pada sistem perangkap positif, perangkap memasukkan carrier interferensi yang termodulasi 2.25 MHz diatas carrier video dari pusat TV kabel pada program sehingga channel tidak dapat disaksikan pada layar TV [4]. Para pelanggan yang menggunakan sistem ini diberikan filter penolakan dengan pita frekuensi yang sempit yang akan melemahkan/mengurangi sinyal/carrier interferensi untuk mengembalikan channel ke kondisi semula. Meskipun trap positif murah, teknik ini mudah untuk dikalahkan. Untuk mendapatkan channel yang asli, yang dibutuhkan hanyalah sebuah filter dengan frekuensi yang tepat dimana frekuensi tersebut mudah untuk diperkirakan.



Gambar 2.1 Bentuk gelombang dari efek trapping [5]

Gambar 2.1 memperlihatkan bentuk gelombang dari efek trapping. fv adalah frekuensi visual sedangkan fw adalah frekuensi *wobbulated*. Trapping tersebut lalu menggeser frekuensinya (*FSK*) selama modulasi frekuensi untuk memfilter frekuensi trapping pada satu sisi saja. Dengan minimum deviasi frekuensi sebesar 0.75 MHz untuk menghindari interferensi frekuensi [5].

Sebuah perangkap dipasang secara seri melalui input *female* dan output *male* dari F- *connector*. Perusahaan kabel TV dapat mencegah pelanggannya mendapatkan siaran dengan cara memasang perangkap yang ”tepat” pada lokasi diluar rumah pelanggan. Dalam hal ini, ”tepat” yang dimaksud adalah mengeliminasi channel-channel yang belum dibayar. Berikut adalah jaringan TV kabel dengan sistem traps :



Gambar 2.2 Diagram sirkuit *trapping* [5]

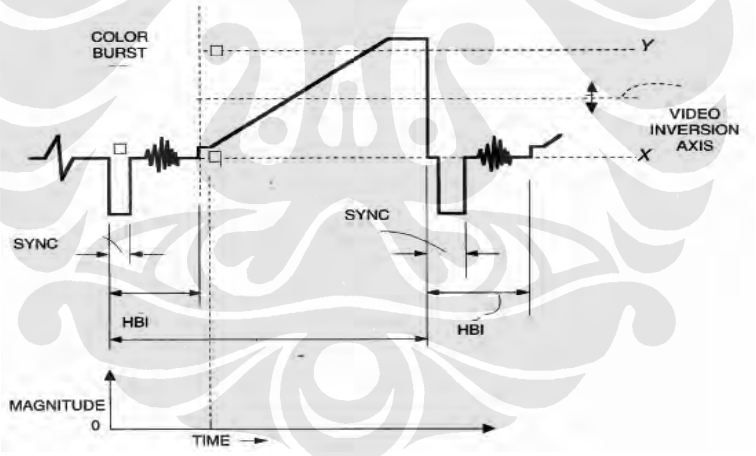
Menyambung dam memutus sebuah perangkap memakan banyak waktu dan sangat tidak efektif. Hal ini dikarenakan si-pemasang kabel (*cable installer*)

harus membuat telepon layanan terlebih dahulu baik ke perusahaan TV kabel maupun ke pelanggan.

2. PEMBALIKKAN VIDEO (*VIDEO INVERSION*)

Metode yang lain, tapi masih digolongkan dengan tingkat keamanan yang rendah untuk enkoding sinyal TV kabel adalah dengan pembalikkan video (*video inversion*). Caranya cukup sederhana. Pola tegangan dari sinyal video dibalik sehingga penerima TV yang lain tidak dapat menginterpretasi sinyal informasi tersebut.

Pembalikkan video secara dinamis sering dipakai untuk metode dasar dari enkripsi video. Sinyal video secara acak dibalik dari suatu frame ke frame yang lain. Sebuah sirkuit, gerbang (*gate*), digunakan untuk memilih mode antara normal atau terbalik. Hal tersebut dibutuhkan dalam proses dekoding sinyal informasi tersebut. Gambar 2.3 memperlihatkan penyederhanaan dari *video inversion.*

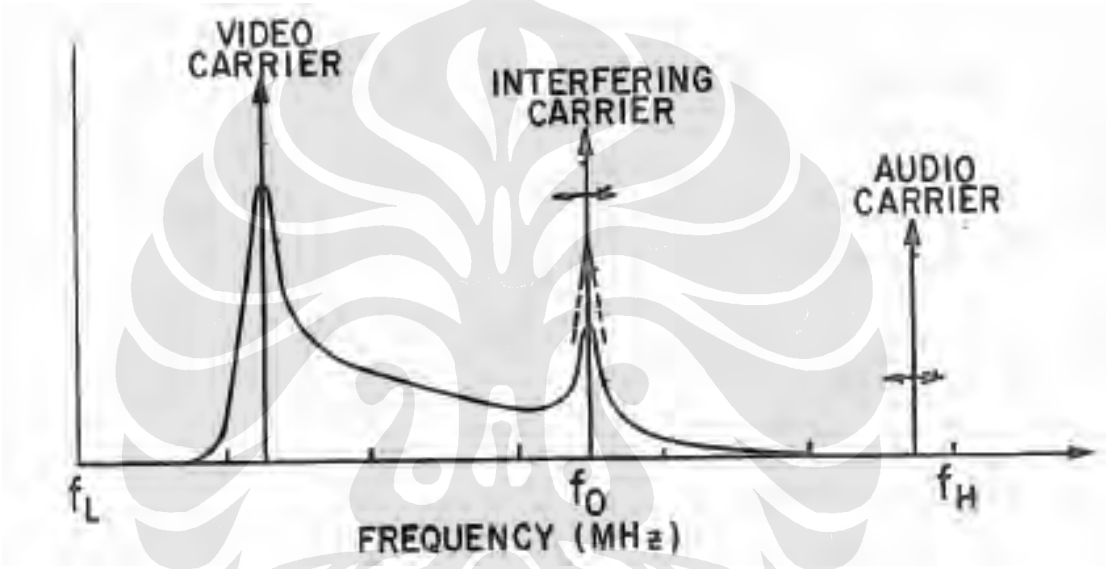


Gambar 2.3 Penyederhanaan video scan line [6]

Terlihat pada Gambar 2.3. bahwasanya pembalikkan video dirancang untuk tidak mempengaruhi pulsa sinkronisasi. Tapi metode ini mempengaruhi pulsa sinkronisasi warna. Hal ini akan menyebabkan perbedaan fasa pada pulsa sinkronisasi warna sebesar 180^{o} sehingga menyulitkan penerima (*receiver*) untuk merekonstruksi sinyal dari 3 warna dasar (*red, green, blue*) [6].

3. CARRIER INTERFENSI (*INTERFERING CARRIER*)

Salah satu teknik *scrambling* video, tapi masih tergolong dengan tingkat keamanan yang rendah adalah dengan menambahkan sebuah carrier interferensi (*interfering carrier*) pada frekuensi dalam channel TV. Derau (*noise*) dengan *bandwidth* yang sempit (bentuknya hampir menyerupai paku) dimasukkan ke ruang frekuensi antara carrier video dan audio sebelum sinyal video dimodulasi ke carriernya. Efek nyata dari interferensi ini dimulai sekitar **40 dB** [4] dibawah carrier video. Hal ini akan mengakibatkan gambar hilang/musnah ketika sinyal scrambling sama dengan level carrier video. Berikut adalah plot frekuensinya :



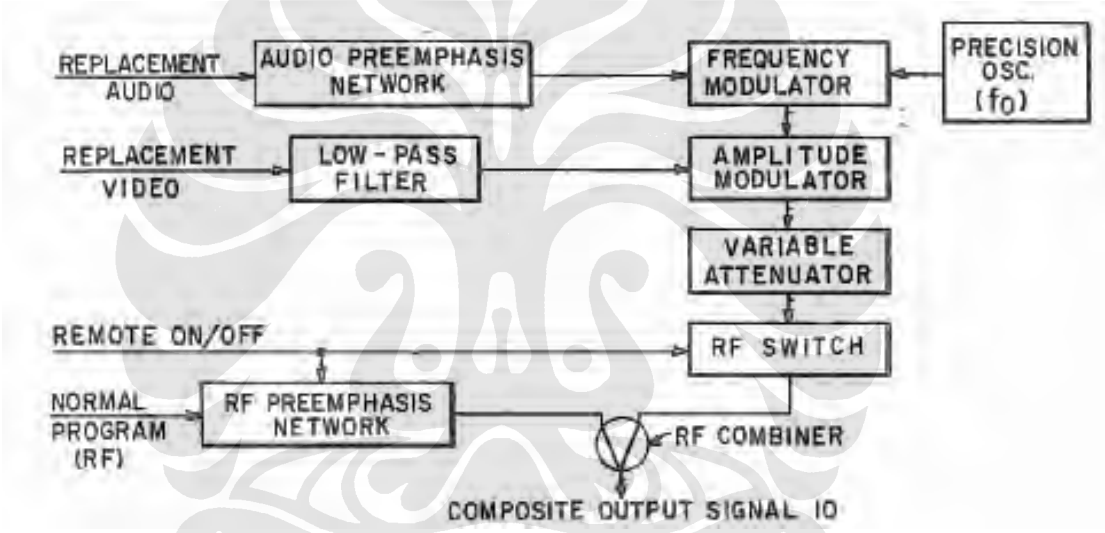
Gambar 2.4 Plot spektrum frekuensi [7]

Terlihat pada Gambar 2.4. bahwasanya sinyal TV tercakup dalam rentang frekuensi rendah (f_L) dan frekuensi tinggi (f_H), yaitu sebesar 6 MHz. Sesuai dengan perjanjian bahwa carrier untuk video terletak pada 1.25 MHz diatas f_L dan carrier untuk audio terletak pada 5.75 MHz diatas f_L [6]. Sebuah sinyal carrier dimodulasikan dengan informasi pengganti. Interfering carrier dimodulasi secara AM dengan carrier video dan dimodulasikan secara FM dengan carrier audio.

Jumlah distorsi yang dihasilkan akan bergantung pada frekuensi dari sinyal interferensi. Sinyal video dan audio akan menjadi lebih rusak/hancur seiiring dengan menurunnya pemisahan frekuensi antara sinyal interferensi dengan sinyal carrier. Frekuensi *scrambling* merupakan perkalian dari frekuensi *horizontal scanning*. Hal ini akan menyebabkan kerusakan yang lebih dibandingkan dengan

sinyal interferensi yang lain. Rahasia dari carrier interferensi jenis ini terletak pada sebuah sinyal 1 kHz yang mengakibatkan garis horizontal muncul pada layar TV [4].

Selain itu, harmonisasi dari 15 kHz sinyal interferensi akan mempengaruhi *vertical scanning* TV sehingga mengakibatkan gambar berputar/*roll*[4] . Kedua sinyal frekuensi tersebut, 1 kHz dan 15 kHz, akan merusak *Automatic Gain Control* (AGC) dan sirkuit warna. Oleh karena itu, sebagian besar sistem interferensi dengan metode ini adalah dengan menggabungkan kedua sinyal 1 kHz dan 15 kHz, bersama untuk memodulasi carrier interferensi. Berikut adalah blok diagram dari proses tersebut :



Gambar 2.5 Blok diagram *carrier interferensi* [7]

Terlihat pada Gambar 2.5. *replacement* audio dan video ini digunakan untuk menghasilkan interfering carrier. Pada *replacement* video terdapat *low-pass* filter. Filter ini di-*setting* agar mempunyai frekuensi *cut-off* lebih rendah dari sinyal video aslinya [6]. Sehingga spektrum AM dari interfering carrier akan memiliki *bandwidth* yang sempit seperti yang telah dibahas. Lalu semua hasil modulasi tersebut akan digabungkan/di-*insert* dengan sinyal aslinya dengan sebuah RF *combiner.*

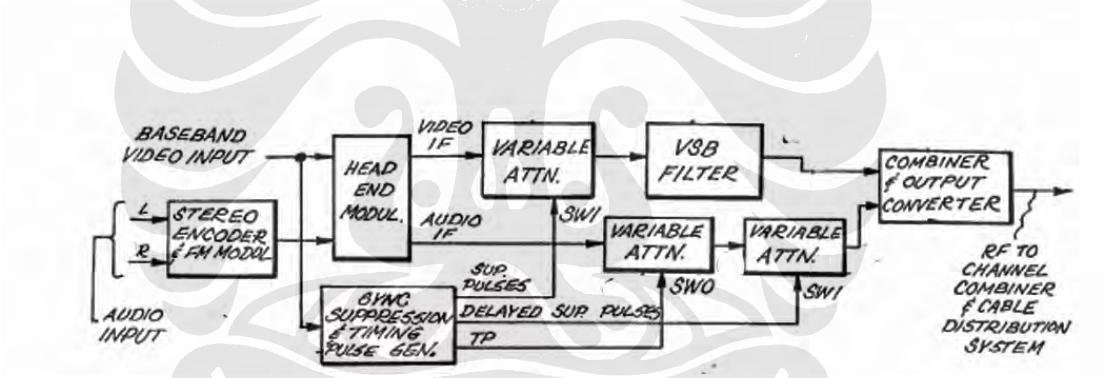
Keuntungan utama dengan menggunakan carrier interferensi adalah biayanya yang rendah. Baik dari sisi *scrambling* maupun peralatan *descrambling* yang digunakan. Bagaimanapun juga, tingkat keamanan dengan metode ini masih

tergolong rendah. Pada kenyataannya, di area-area yang penyiarannya menggunakan carrier interferensi yang relatif lemah dan tidak ada channel yang berbatasan dengan channel yang telah di-*scramble*, "*detuning*" TV dari interferensi membuat seseorang dapat menangkap ulang sinyal video, meskipun hanya hitam putih.

4. HORIZONTAL SYNC SUPPRESSION

Metode ini tergolong dengan tingkat keamanan menengah. Peralatan encoding dan decoding-nya cukup sederhana dan juga tidak terlalu mahal [8].

Ketika pulsa sinkronisasi horizontal ditekan ke level yang lebih rendah dimana penerima TV kehilangan sinkronisasi, gambar TV akan terlihat seperti sekumpulan garis yang bergelombang. Efek ini terjadi karena setiap garis "*scanner*" pada layar TV start pada posisi yang acak. Berikut adalah blok diagram sistem ini :



Gambar 2.6 Blok diagram TV kabel dengan sync suppression [8]

Pada Gambar 2.6. teknik *scrambling* dengan menekan pulsa sinkronisasi berhubungan erat dengan modulator, sebuah alat yang memproses sinyal audio dan video (sinyal komposit) ke sebuah frekuensi dari channel yang bersesuaian. Peralatan enkoding tersebut terus memantau sinyal komposit video dan menghasilkan kumpulan pulsa yang mempunyai urutan perwaktuan yang sama seperti pulsa sinkronisasi horizontal yang asli. Lalu pulsa tersebut ditambahkan ke sinyal aslinya untuk menekan amplitudo pulsa sinkronisasi horizontal. Penurunan tegangan sebesar 6 dB secara efektif menurunkan tegangan sinkronisasi dibawah level hitam dari TV ("*Black Reference Level*") ke tegangan dimana sinyal informasi ditransmisikan [8].

Kerusakan gambar akan berlanjut. Hal ini dikarenakan menekan pulsa sinkronisasi horizontal akan membingungkan ”Automatic Gain Control” (AGC) [4]. Penerima TV menggunakan pulsa sinkronisasi horizontal sebagai referensi tegangan maksimum. Ini akan membuat amplifier menyesuaikan ulang penguatan (gain) sebagaimana sinyal yang dipancarkan bervariasi. Sehingga level tegangan yang memancarkan sinyal informasi gambar pada setiap garis scan akan berada pada interval antara level tegangan hitam dan level tegangan putih. Ini akan menjamin akan adanya batas maksimum untuk hitam dan putih. Tanpa sinkronisasi horizontal sebagai referensi untuk AGC, sirkuit TV akan menduga bahwa level tegangan tertinggi pada sinyal informasi terletak pada level hitam [4]. Sebagai konsekuensinya, gambar yang bukan hitam, yang memiliki level tegangan yang relatif tinggi akan diinterpretasikan sebagai hitam sempurna.

Menekan pulsa sinkronisasi horizontal mempunyai kerugian dalam hal penurunan perbandingan sinyal video terhadap ”noise” (derau). Penurunan level video ini terjadi dengan rasio yang sama seperti penekanan pulsa sinkronisasi [8]. Hal ini terjadi karena sinyal informasi harus diturunkan dengan rasio yang sama sebagaimana level sinkronisasi ditekan untuk menjamin tercapainya rekonstruksi gambar.

Akhir-akhir ini banyak operator kabel yang terganggu dengan adanya ”kotak hitam” atau dekoder yang tidak terlisensi. Banyak pelanggan kabel TV yang berencana untuk menyaksikan program-program premium tanpa membayar tagihan bulanan. Oleh karena itu, dasar keamanan dari penekanan pulsa sinkronisasi telah ditingkatkan dengan menambahkan banyak level sehingga menyulitkan untuk menentukan level yang tepat untuk rekonstruksi sinyal informasi. Pulsa dekoding yang ditumpangkan pada sub-carrier audio juga telah digeser perwaktuannya. Bahkan informasi pengalamatan ditumpangkan pada sub-carrier yang terpisah.

Hampir semua dekoder untuk metode ini bekerja pada frekuensi channel 3 atau 4 [8]. hal ini dimaksudkan dalam hubungannya dengan alat pengubah (converter) pada standar kabel TV. Pada kenyataannya, beberapa merek dekoder tidak bekerja sebagaimana mestinya. Dekoder tersebut akan bekerja sampai cocok dengan converter yang bersesuaian.

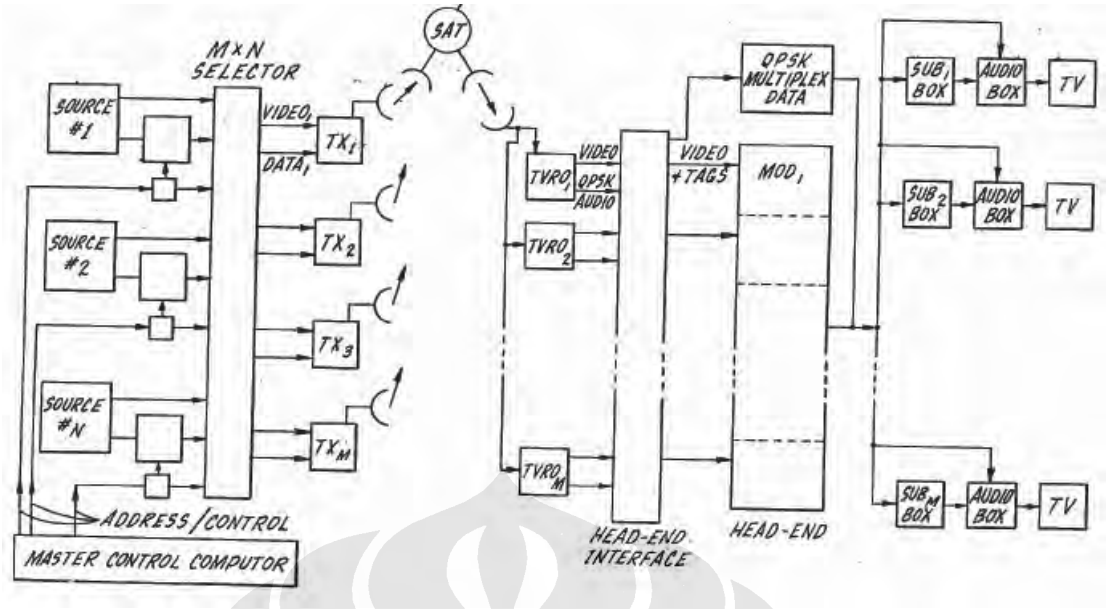
5. SYNC REMOVAL

Penghilangan pulsa sinkronisasi secara keseluruhan adalah metode yang lebih aman lagi dibandingkan dengan menekan atau menggeser perwaktuan dari sinyal informasi. Kunci untuk men-decode metode ini terletak pada rekonstruksi pulsa sinkronisasi dan memasukkan pulsa tersebut ke lokasi yang tepat di dalam sinyal komposit video.

Metode ini di-decode dengan sirkuit yang dapat membangkitkan pulsa sinkronisasi yang dibutuhkan oleh sinyal TV secara penuh. Sirkuit ini dapat mensinkronisasi sinyal yang akan datang dengan cara mendeteksi ”built in clock” sinyal yang ditransmisikan bersamaan dengan data. Lalu mengeset pemicu dari pembangkit pulsa pada poin yang sesuai. Pembangkit sinkronisasi bekerja secara kontinu dan output-nya secara tepat ditambahkan ke sinyal video, yang pulsa sinkronisasinya telah dihilangkan, selama interval sinkronisasi horizontal. Sebuah tegangan DC referensi yang didapatkan dari pulsa pada interval ”blanking” secara vertikal, digunakan untuk mengeset rekonstruksi pulsa sinkronisasi horizontal ke level yang tepat.

6. DIGITAL AUDIO ENCRYPTION

Enkripsi dari pemancaran satelit haruslah sangat rahasia/aman, dengan mempertimbangkan banyaknya penonton dan potensi untuk pembajakan. Sistem yang dipilih untuk mengatasi masalah ini adalah teknik *scrambling* yang dilakukan baik audio maupun video. Berikut adalah block diagram dari teknik tersebut :



Gambar 2.7 Diagram sistem distribusi TV kabel

Dengan *DA Encrypter*[9]

Gambar 2.7. memperlihatkan kemajemukan sumber dari sinyal CATV, mulai dari sumber ke -1 sampai ke-N. Sebuah *master* kontrol komputer akan menyediakan data kontrol untuk proses *descrambling*. Komputer ini terhubung dengan data alamat lalu dimasukkan ke prosesor audio dimana sinyal audio akan terenkripsi. Tidak hanya sinyal audio saja yang terenkripsi tapi juga data kontrol dan informasi pengalamatan (*unlocking key*) juga terenkripsi. Setiap sinyal CATV akan tehubung dengan M x N selector. Lalu semua sinyal tersebut ditransmisikan dengan antena ke sebuah satelit. Satelit berfungsi sebagai jaringan komunikasi antara sisi pengirim dan sisi penerima.

Pada sisi penerima, sebuah sinyal digital ditangkap oleh dekoder dan dimasukkan ke sebuah tempat penyimpanan sementara yang disebut ”*shift register*” [9] . Lalu diproses kedalam sebuah mikrokomputer. Metode ini di-*decsramble* dengan cara membalik kode algoritma yang digunakan. Sinyal informasi ini akan kembali ke bentuk aslinya sehingga video dan audio dapat dipisahkan.

Kunci pembuka (*unlocking key*) juga ditransmisikan bersamaan dengan sinyal informasi dan harus secara terpisah agar dapat mengaktifkan dekoder dan membalik algoritma dari *scrambling*. Perlu diketahui bahwa kunci tersebut tidak

mengandung semua informasi pembuka. Tapi semata-mata sebagai katalisator yang membuka daya/power dari rangkaian dekoding.

Sistem ini mempunyai fleksibilitas yang baik [9]. Pengalamatan dari masing-masing pelanggan berada pada basis yang dinamis. Hal ini dikarenakan komputer pusat dapat men-*cycle* melalui daftar layanan pelanggan dan men-*download* lisensi informasi. Setiap terminal pada sistem menerima semua informasi tapi hanya merespon pada data yang dialamatkan kepada sistem tersebut [10]. Satu dekoder dapat digunakan untuk membuka *scrambling* channel manapun ketika informasi pengalamatan yang digabungkan dengan lisensi informasi yang tepat ditransmisikan dari komputer kontrol.

2.2. SISTEM OPERASI MATRIKS [16]

2.2.1 Definisi

Matriks adalah susunan bilangan yang diatur berdasarkan baris dan kolom. Susunan bilangan yang terdapat dalam matriks tersebut biasanya di tuliskan dalam sebuah tanda kurung kotak [.... ....] yang sesuai dengan dimensi matriks. Dimensi/ordo/ukuran suatu matriks ditentukan oleh banyaknya jumlah baris diikuti oleh banyaknya jumlah kolom. Contoh :

$$A_{2 \times 3}$$

Artinya : matriks A mempunyai 2 baris dan 3 kolom

Jika susunan bilangan dari matriks A ditulis, maka akan terlihat seperti :

$$\begin{pmatrix} A_{11} & A_{12} & A_{13} \\ A_{21} & A_{22} & A_{23} \\ A_{31} & A_{32} & A_{33} \end{pmatrix}$$

2.2.2. Kesamaan Matriks

Ditulis **A = B**

Dua buah matriks dikatakan sama, jika :

1.Ordonya sama

2.Elemen-elemen yang seletak sama

2.2.3. Matriks Transpose

Ditulis $\mathbf{A^t}$ **;** $\mathbf{A'}$ **;** $\overline{\mathbf{A}}$

Matriks transpose adalah matriks yang elemen-elemen barisnya adalah elemen-elemen kolom **A** sedangkan elemen-elemen kolomnya adalah elemen-elemen baris **A**.

2.2.4. Aljabar Matriks [17]

2.2.4.1. Penjumlahan dan Pengurangan

Ditulis **A + B** dan **A – B**

Penjumlahan atau pengurangan dua matriks **A** dan **B** adalah matriks yang didapat dengan menjumlahkan atau mengurangkan setiap elemen-elemen **A**

dengan elemen-elemen **B** yang bersesuaian. Hal ini dapat dilakukan dengan syarat matriks **A** dan **B** harus berordo sama. Contoh :

$$\begin{pmatrix} A_{11} & A_{12} & A_{13} \\ A_{21} & A_{22} & A_{23} \\ A_{31} & A_{32} & A_{33} \end{pmatrix} + \begin{pmatrix} B_{11} & B_{12} & B_{13} \\ B_{21} & B_{22} & B_{23} \\ B_{31} & B_{32} & B_{33} \end{pmatrix} = \begin{pmatrix} A_{11}+B_{11} & A_{12}+B_{12} & A_{13}+B_{13} \\ & \text{dst} & \\ & & \end{pmatrix}$$

2.2.4.2. Perkalian Dua Matriks

Dua matriks **A** dan **B** terdefinisi untuk dikalikan. Hal ini dapat dilakukan jika banyaknya kolom **A** = banyaknya baris **B**. Dengan hasil suatu matriks C yang berukuran $_{\text{baris A x kolom B}}$. Contoh :

$$\mathbf{A_{m\,x\,n} \cdot B_{n\,x\,p} = C_{m\,x\,p}}$$

Jika elemen bilangannya dituliskan, maka akan seperti :

$$\begin{pmatrix} A_{11} & A_{12} & A_{13} \\ A_{21} & A_{22} & A_{23} \\ A_{31} & A_{32} & A_{33} \end{pmatrix} \times \begin{pmatrix} B_{11} & B_{12} & B_{13} \\ B_{21} & B_{22} & B_{23} \\ B_{31} & B_{32} & B_{33} \end{pmatrix} = \begin{pmatrix} (A_{11}.B_{11} + A_{12}.B_{21} + A_{13}.B_{33}) & \dots & \dots \\ (A_{21}.B_{11} + A_{22}.B_{21} + A_{23}.B_{31}) & \text{dst} & \\ (A_{31}.B_{11} + A_{32}.B_{21} + A_{33}.B_{31}) & \dots & \dots \end{pmatrix}$$

Mengalikan baris-baris **A** dengan kolom-kolom **B** kemudian menjumlahkan hasil perkalian tersebut.

2.2.4.3. Perkalian Matriks dengan Skalar

Ditulis **k A**

Jika **k** suatu skalar dan **A** suatu matriks, maka **kA** adalah matriks yang diperoleh dengan mengalikan setiap elemen **A** dengan **k.**

2.2.5. Jenis-Jenis Matriks

Ada beberapa jenis matriks yang akan dibahas dalam dasar teori skripsi ini. Diantaranya adalah :

1. Matriks nol

Adalah matriks yang semua elemennya adalah angka 0.

$$\mathbf{A} = \begin{pmatrix} 0 & 0 \\ 0 & 0 \end{pmatrix} \quad \text{atau} \quad \mathbf{B} = \begin{pmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix}$$

2. Matriks satuan

Adalah matriks bujur sangkar yang semua elemen diagonal utamanya adalah 1. Sedangkan elemen lainnya adalah 0. matriks satuan diberi notasi : I (**identitas**).

$$\mathbf{A} = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} \quad \text{atau} \quad \mathbf{B} = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

Sifat : AI = IA = A

3. Matriks segitiga atas atau matriks segitiga bawah

Matriks segitiga atas adalah matriks bujur sangkar yang semua elemen dibawah diagonal utamanya adalah 0. Sedangkan matriks segitiga bawah adalah kebalikan dari matriks segitiga atas.

$$\mathbf{A} = \begin{pmatrix} a & b & c \\ 0 & e & f \\ 0 & 0 & i \end{pmatrix} \quad \text{atau} \quad \mathbf{B} = \begin{pmatrix} a & 0 & 0 \\ d & e & 0 \\ g & h & i \end{pmatrix}$$

Keterangan : **A =** matriks segitiga atas

B = matriks segitiga bawah

4. Matriks diagonal

Adalah matriks bujur sangkar yang semua elemen selain elemen diagonal utamanya adalah 0.

$$\mathbf{A} = \begin{pmatrix} a & 0 \\ 0 & d \end{pmatrix} \quad \text{atau} \quad \mathbf{B} = \begin{pmatrix} a & 0 & 0 \\ 0 & e & 0 \\ 0 & 0 & i \end{pmatrix}$$

2.2.6. Determinan Matriks

Determinan matriks adalah nilai atau besar dari matriks itu sendiri. Agar memperoleh determinan dari matriks, maka matriks harus berbentuk bujur sangkar. Perhitungan determinan matriks untuk masing-masing ordo, menggunakan metode yang berbeda. Dalam dasar teori ini, hanya akan dibatasi untuk matriks ordo 3 x 3. Notasi determinan ditulis dengan : **|A| atau det A**

1.Ordo 2

$$\mathbf{A} = \begin{pmatrix} a & b \\ c & d \end{pmatrix}$$

$$|\mathbf{A}| = ad - bc$$

2.Ordo 3

$$\mathbf{A} = \begin{pmatrix} a & b & c \\ d & e & f \\ g & h & i \end{pmatrix}$$

|A| :

a. Ekspansi baris pertama

$$|\mathbf{A}| = a\begin{vmatrix} e & f \\ h & i \end{vmatrix} - b\begin{vmatrix} d & f \\ g & i \end{vmatrix} + c\begin{vmatrix} d & e \\ g & h \end{vmatrix}$$

b. Aturan Sarrus

$$|\mathbf{A}| = \begin{vmatrix} a_1 & b_1 & c_1 \\ a_2 & b_2 & c_2 \\ a_3 & b_3 & c_3 \end{vmatrix} \begin{matrix} a_1 & b_1 \\ a_2 & b_2 \\ a_3 & b_3 \end{matrix} =$$

$$|\mathbf{A}| = a_1b_2c_3 + b_1c_2a_3 + c_1a_2b_3 - a_3b_2c_1 - b_3c_2a_1 - c_3a_2b_1$$

2.2.7. Matriks Invers

Jika **A** dan **B** adalah matriks bujur sangkar dengan ordo dua atau lebih dan **AB = BA = I.** Maka **B** dikatakan *invers* dari **A** (ditulis $\mathbf{A^{-1}}$) dan **A** dikatakan invers dari **B** (ditulis $\mathbf{B^{-1}}$). Berikut adalah contoh untuk yang berordo 2 :

$$\mathbf{A} = \begin{pmatrix} a & b \\ c & d \end{pmatrix} \quad \rightarrow \quad \mathbf{A^{-1}} = \frac{1}{ad - bc} \begin{pmatrix} d & -b \\ -c & a \end{pmatrix}$$

Matriks A dikatakan mempunyai *inver* jika $|\mathbf{A}| \neq \mathbf{0}$ dan disebut matriks *non-singular.*

Jika $|\mathbf{A}| = \mathbf{0}$, maka **A** disebut matriks singular.

Sifat : $\mathbf{AA^{-1}} = \mathbf{A^{-1}A} = \mathbf{I}$

2.2.8. Sifat-Sifat

Matriks memiliki beberapa sifat, diantaranya :

1. $(A^t)^t = A$

2. $(A + B)^t = A^t + B^t$

3. $(A \cdot B)^t = B^t \cdot A^t$

4. $(A^{-1})^{-1} = A$

5. $(A \cdot B)^{-1} = B^{-1} \cdot A^{-1}$

6. $|A^t| = |A|$

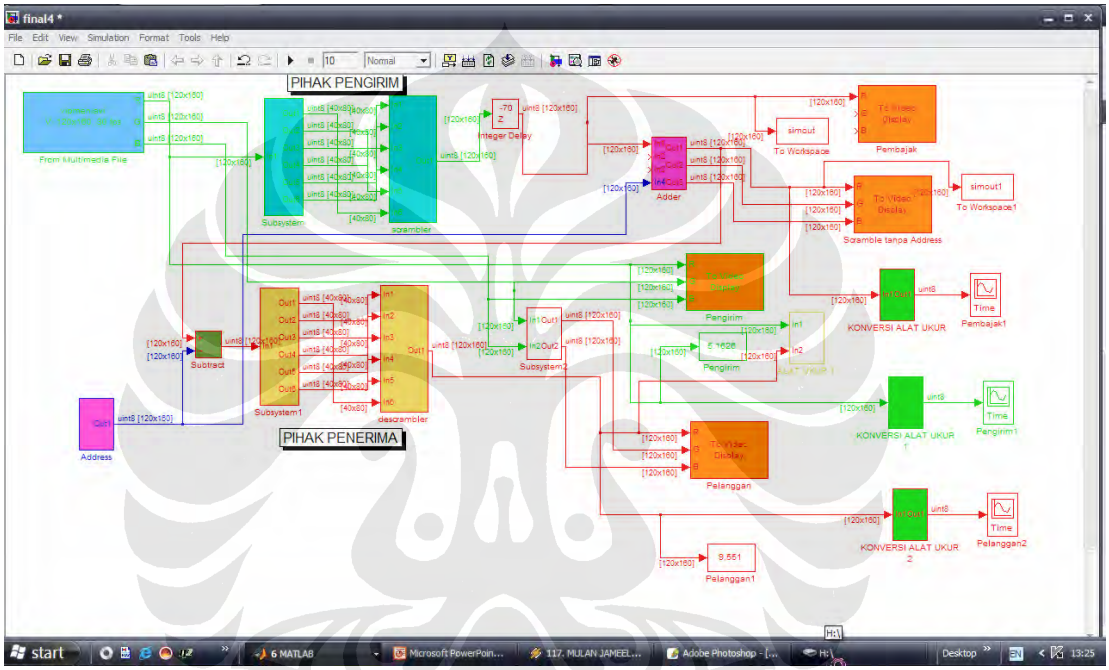
7. $AB = C \rightarrow A = CB^{-1}$

$\rightarrow B = A^{-1}C$

BAB 3
KONFIGURASI VIDEO *SCRAMBLER*

3.1. BAGAN VIDEO *SCRAMBLER*

Simulasi dalam skripsi ini menggunakan Simulink dari MATLAB versi 2006b, dimana skema dari video *scrambler* terlihat seperti yang diilustrasikan pada Gambar 3.1.



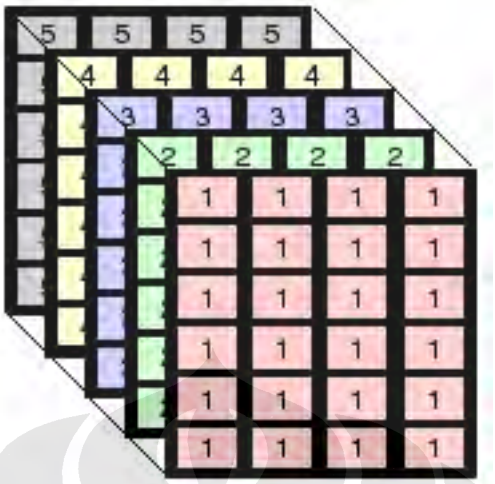
Gambar 3.1. Skema video *scrambler* [18]

Skema dari video *scrambler* yang digunakan dalam simulasi skripsi ini terdiri dari blok-blok diagram yang memiliki fungsi masing-masing. Berikut nama dan fungsi sekaligus konfigurasi dari blok video *scrambler* yang digunakan dalam simulasi skripsi [20] :

1) From Multimedia File

Blok ini berfungsi untuk membaca video frame atau sample audio dari multimedia file yang sudah terkompresi dan mengimpornya ke dalam model simulink. Jumlah pixel video yang dipilih dari blok ini berdimensi 120 x 160 (2D). Akan tetapi pada kenyataannya (setelah melihat hasil keluaran pada *workspace* MATLAB), Blok *from multimedia file* ini memiliki dimensi 4D, yaitu

120x160x301 dan 1 dimensi lagi untuk perwaktuan. Gambar 3.2. memperlihatkan ilustrasinya.



Gambar 3.2. Ilustrasi matriks input video [20]

Port output dari blok ini akan berubah sesuai dengan isi yang dibaca dari file multimedia.Tabel 3.1. memperlihatkan konfigurasi port output dari blok ini.

Tabel 3.1. Konfigurasi port output blok *from multimedia file*

Port	Output	Supported Data Types	Supports Complex Values?
R, G, B	Matrix that represents one plane of the RGB video stream. Outputs from the R, G, or B port must have same dimensions.	• Double-precision floating point • Single-precision floating point • 8-, 16-, and 32-bit signed integers • 8-, 16-, and 32-bit unsigned integers	No
I	Matrix that represents the intensity video stream	• Double-precision floating point • Single-precision floating point • 8-, 16-, and 32-bit signed integers • 8-, 16-, and 32-bit unsigned integers	No
Audio	Vector of audio data	• Double-precision floating point • Single-precision floating point • 16-bit signed integers • 8-bit unsigned integers	No

sumber : simulink library browser

Konfigurasi yang dilakukan pada simulasi :

- *Input file name* → **vipmen.avi**
- *Output* → **video only**
- *Video output data type* → **uint 8**
- *Number of times to play file* → **inf**
- *Output end-of-file indicator*
- *Inherit sample time from file*

Blok *from multimedia file* tidak dapat mendukung semua jenis format file. Tabel 3.2. memperlihatkan jenis format yang dapat didukung oleh blok ini.

Tabel 3.2. Format blok *from multimedia file*

Port	Output	Supported Data Types	Supports Complex Values?
R, G, B	Matrix that represents one plane of the RGB video stream. Outputs from the R, G, or B port must have same dimensions.	• Double-precision floating point • Single-precision floating point • 8-, 16-, and 32-bit signed integers • 8-, 16-, and 32-bit unsigned integers	No
I	Matrix that represents the intensity video stream	• Double-precision floating point • Single-precision floating point • 8-, 16-, and 32-bit signed integers • 8-, 16-, and 32-bit unsigned integers	No
Audio	Vector of audio data	• Double-precision floating point • Single-precision floating point • 16-bit signed integers • 8-bit unsigned integers	No

Sumber : simulink library browser

Konfigurasi untuk *video output data type* dipilih bilangan *uint 8* (unsigned integer 8) karena tidak ada bilangan negatif (-…) pada *uint 8*. Selain itu, bilangan *uint 8* merupakan bilangan dengan panjang terpendek dibandingkan dengan yang lainnya (*single, double, int 16*). Hal ini akan memudahkan dalam menjalankan proses simulasi.

2) To Video Display

Blok ini berfungsi untuk mengirimkan data video ke sebuah video output atau camera video yang didukung dengan DirectX. Blok ini dapat mengirimkan data video ke monitor yang berbeda atau menampilkannya pada sebuah window dari monitor. Tabel 3.3. memperlihatkan konfigurasi port input dari blok ini.

Tabel 3.3. Konfigurasi port input blok *to video display*

Port	Input	Supported Data Types	Supports Complex Values?
I	Matrix of intensity values	• Double-precision floating point • Single-precision floating point • Boolean • 8-, 16, and 32-bit signed integers • 8-, 16, and 32-bit unsigned integers	No
R, G, B	Matrix that represents one plane of the RGB video stream. Inputs to the R, G, or B ports must have the same dimensions and data type.	Same as I port	No

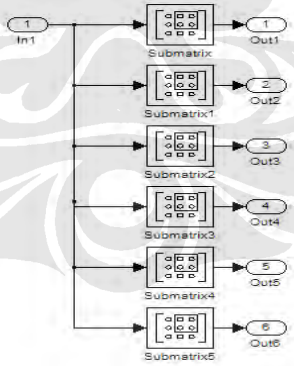
Sumber : simulink library browser

Konfigurasi dalam blok ini :

- *Input image type* → **RGB**
- *Video output device* → **on-screen video monitor**

3) Subsystem dan *subsystem* 1

Blok ini berfungsi untuk membagi pixel dari blok *from multimedia file*. Isi dari blok *subsystem* itu sendiri hanyalah sekumpulan dari blok *sub-matrix*. Gambar 3.2. menunjukkan skema dari blok *subsystem*.



Gambar 3.3. Isi dari blok *subsystem* [18]

a) Submatrix

Blok *submatrix* akan mengekstrak/membagi pixel video dari blok *from multimedia file*. Jumlah pixel 120x160 sebenarnya disusun dalam bentuk matriks yang ditransmisikan pada setiap *frame*. Masing-masing elemennya mewakili intensitas atau warna tertentu dari video. Terlihat pada Gambar 3.3. terdapat 6

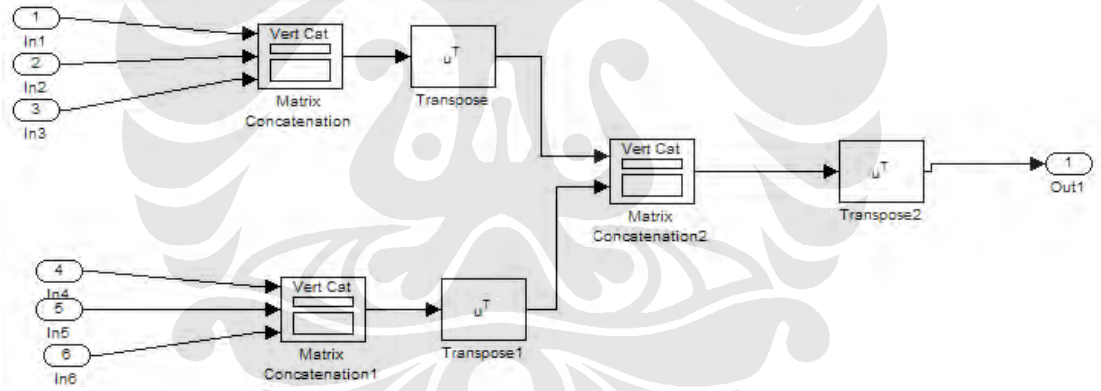
blok *submatrice*. Hal ini berarti pixel video akan dibagi menjadi 6 sekumpulan bagian pixel saja. Seperti terlihat pada Tabel 3.4.

Tabel 3.4. Pembagian *pixel* video

	1	**80**	**160**
		Pixel 1	*Pixel* 4
40		*Pixel* 2	*Pixel* 5
80		*Pixel* 3	*Pixel* 6
120			

4) Scrambler/descrambler

Blok ini bukan berfungsi seperti blok *scrambler/descrambler* yang terdapat pada *communication blockset* → *sequence operation* → *scrambler/descrambler*. Akan tetapi, blok ini diinisialisasi sendiri oleh penulis. Isi dari Blok *scrambler/descrambler* ini adalah sebuah subsistem yang terdiri dari *matrix concatenation* dan *matrix transpose* dan juga manipulator-manipulator matriks lainnya. Gambar 3.4. memperlihatkan isi dari blok ini.



Gambar 3.4. Subsistem dari blok *scrambler/descrambler* [20]

a) *Matrix concatenation*

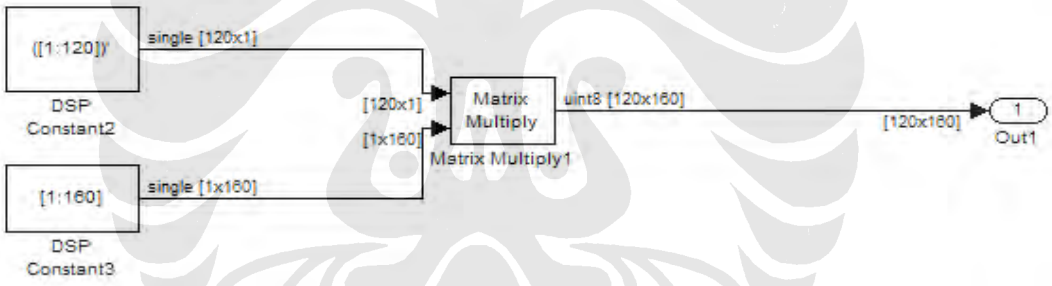
Blok *concatenate* berfungsi menggabungkan sinyal input dengan tipe data yang sama, untuk menghasilkan sebuah *output* sinyal yang elemennya terletak bersebelahan dalam memori. Blok ini beroperasi dalam vektor atau *matrix concatenation mode* yang tergantung pada pengaturan parameter *mode* (horizontal atau vertikal). Oleh karenanya, input dapat digabungkan dari atas ke bawah atau dari kiri ke kanan pada sisi input. Pada simulasi ini digunakan metode ***vertical.***

b) *Matrix transpose*

Blok transpose seperti yang sudah dijabarkan pada Bab 2, berfungsi untuk membalik baris pada matriks input → kolom pada matriks output dan kolom pada matriks input → baris pada matriks output. Hal ini dilakukan agar dimensi/ordo/ukuran dari matriks yang pixel-pixelnya telah dibagi menjadi sama seperti inputan video dari blok *from multimedia file,* yaitu 120 x 160.

5) *Address*

Blok ini berfungsi sebagai alamat yang dimiliki oleh pelanggan ataupun sebagai ”kunci pembuka” video . Prinsip dari alamat ini sama dengan prinsip pengiriman *e-mail* pada internet. Masing-masing pelanggan dan video akan diberi alamat unik yang dimiliki oleh satu pelanggan dan untuk satu jenis video saja. Pada simulasi ini, blok *address* sebenarnya adalah sekumpulan dari blok *DSP constant.* Gambar 3.5. memperlihatkan subsistem dari *Address* tersebut.



Gambar 3.5. Subsistem dari blok *address* [20]

a) *DSP Constant*

Blok ini berfungsi untuk membangkitkan sebuah sinyal dimana nilainya tetap terjaga konstan selama proses simulasi berlangsung. Nilai konstan tersebut dapat berupa skalar, vektor ataupun matriks.

Konfigurasi yang dilakukan pada simulasi :

- *Constant value* → **([1:120])’ dan ([1:160])**
- *Sample mode* → **discrete**
- *Output* → **sample-based**

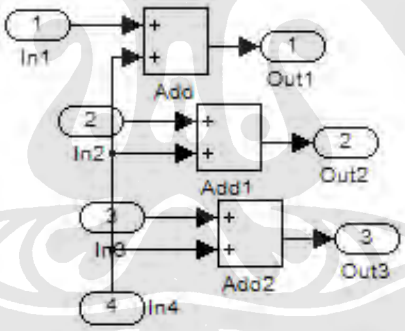
Elemen matriks yang terbentuk ketika dituliskan ([1:120])' adalah angka yang berurutan mulai dari 1, 2, 3, ..., 120 secara vertikal (matriks kolom). Sedangkan ([1:160]) akan membentuk angka yang berurutan mulai dari 1, 2, 3, ..., 160 secara horizontal (matriks baris). Outputnya akan berupa 2 matriks *sample* yang berukuran 120 x 1 dan 1 x 160.

b) *Matrix multiply*

Blok ini berfungsi untuk mengalikan elemen-elemen matriks yang telah terdefinisi pada *DSP constant*. Hasil dari *matrix multiply* ini adalah sebuah sekumpulan bilangan dengan dimensi yang sama dari pixel video sumber, 120 x 160.

6) Adder/Substract

Blok ini berfungsi untuk menjumlahkan/mengurangkan elemen-elemen matriks antara matriks R dengan matriks *Adder/substract.* Blok ini terdiri dari subsistem seperti yang diperlihatkan pada Gambar 3.6.



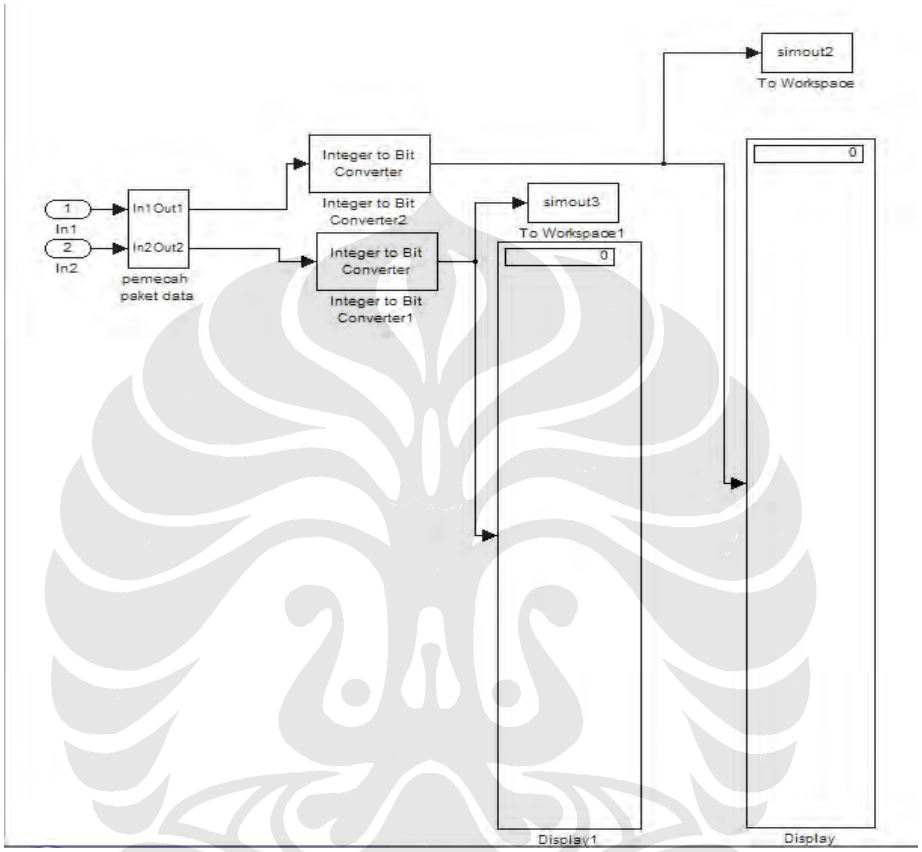
Gambar 3.6. Subsistem dari blok *adder/substract* [20]

Seperti yang sudah dijelaskan pada Bab 2, bahwasanya penjumlahan matriks dapat berlangsung jika dan hanya jika dimensi/ordo/ukuran dari matriks yang berkaitan nilainya sama. Berikut adalah konfigurasinya :

- In 1 → input matriks dari blok *Scrambler* (120 x 160)
- In 4 → Input matriks *address* dari blok *address* (120 x 160)
- Sedangkan In 2 dan 3 tidak dihubungkan dengan apapun dan berfungsi sebagai port tambahan saja.

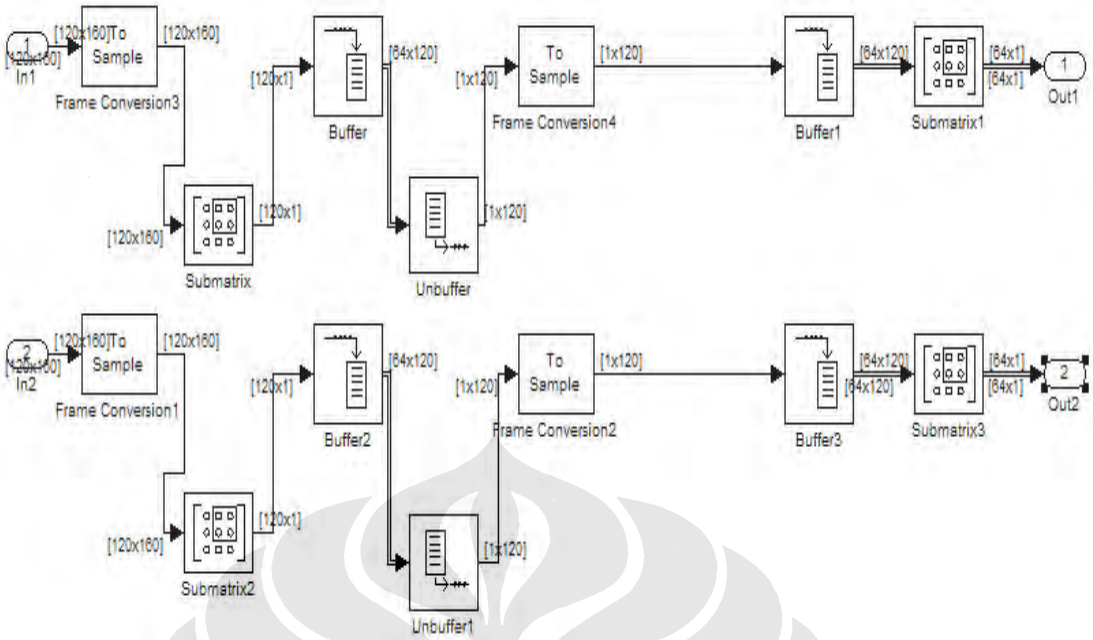
7) Alat Ukur 1

Blok ini berfungsi untuk memperlihatkan banyak bit yang dibangkitkan dari *sample* video yang dipilih sekaligus memperlihatkannya ke dalam *workspace* MATLAB. Blok ini terdiri dari subsistem seperti yang diperlihatkan pada Gambar 3.7.



Gambar 3.7. Subsistem dari blok *Alat Ukur 1*[20]

Terlihat pada Gambar 3.7. subsistem terdiri dari blok *workspace, integer to bit converter* dan *display*. Akan tetapi, sebelum konfigurasi 3 blok tersebut, blok subsistem *alat ukur 1* juga mempunyai sub-subsistem lagi, yaitu blok *pemecah paket data.* Gambar 3.8. memperlihatkan subsistem dari blok *pemecah paket data*.



Gambar 3.8. Subsistem dari blok *pemecah paket data* [18]

Terlihat pada Gambar 3.8., blok *pemecah paket data* terdiri dari *frame conversion, submatrix, buffer* dan *unbuffer*. Seperti yang telah kita ketahui bahwa input video berdimensi 4D. Semua *display* alat pengukur yang ada pada MATLAB versi 2006b tidak dapat mengukur inputan dalam full matriks atau tepatnya 524.288 elemen matriks. Oleh karenanya, agar dapat diukur, matriks input video harus dipecah-pecah dalam paket data.

a) Pemecah Paket Data

i. Frame Conversion

Blok *frame conversion* berfungsi untuk menspesifikasi status dari frame dari sinyal output. Parameter sinyal output dari blok ini ada 2, yaitu *sample based* atau *frame based*. Blok ini tidak *re-buffer* atau mengatur ulang ukuran input. Blok *frame conversion* ini dapat mendukung sejumlah format data seperti yang ditunjukkan pada Tabel 3.5.

Tabel 3.5. Format data blok *frame conversion*

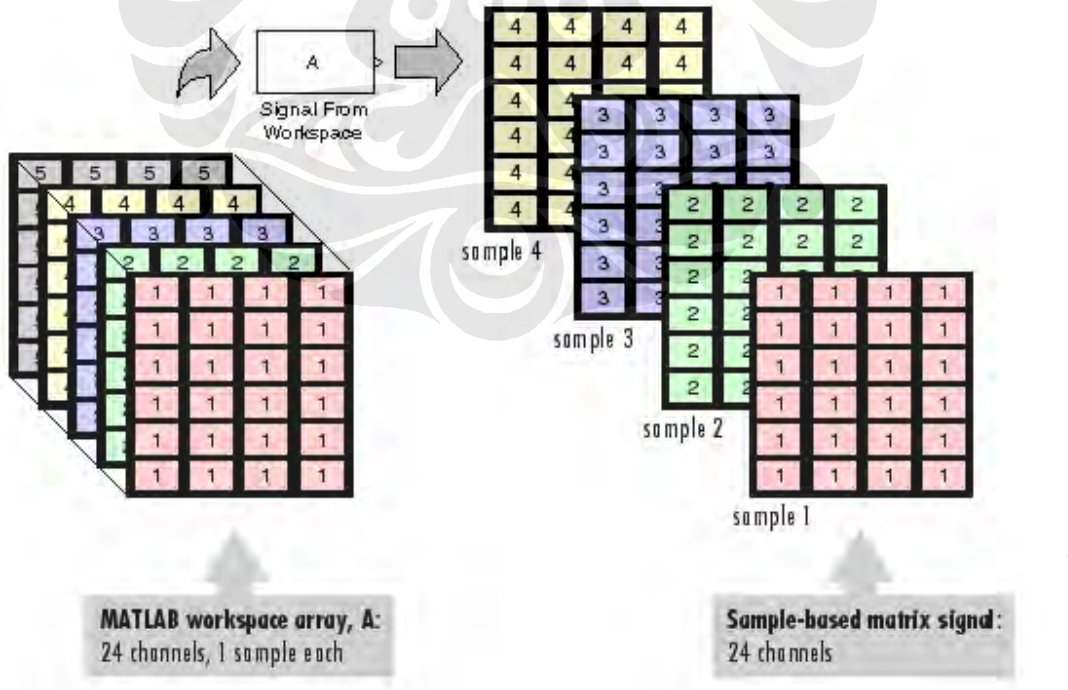
Port	Supported Data Types
Input	• Double-precision floating point • Single-precision floating point • Fixed point (signed only) • Custom data types • Boolean • 8-, 16-, and 32-bit signed integers • 8-, 16-, and 32-bit unsigned integers
Output	• Double-precision floating point • Single-precision floating point • Fixed point (signed only) • Custom data types • Boolean • 8-, 16-, and 32-bit signed integers • 8-, 16-, and 32-bit unsigned integers

Sumber : signal processing blockset-categorial list

Konfigurasi pada *dialog box* :

- *Output signal* → **sample based**

Input video yang terdiri dari 4-D elemen, masing-masing matriksnya akan di-*sample* /di-ekstrak menjadi sekumpulan matriks berjajar (*array*) yang berdimensi 120x160. Gambar 3.9. menunjukkan ilustrasinya.

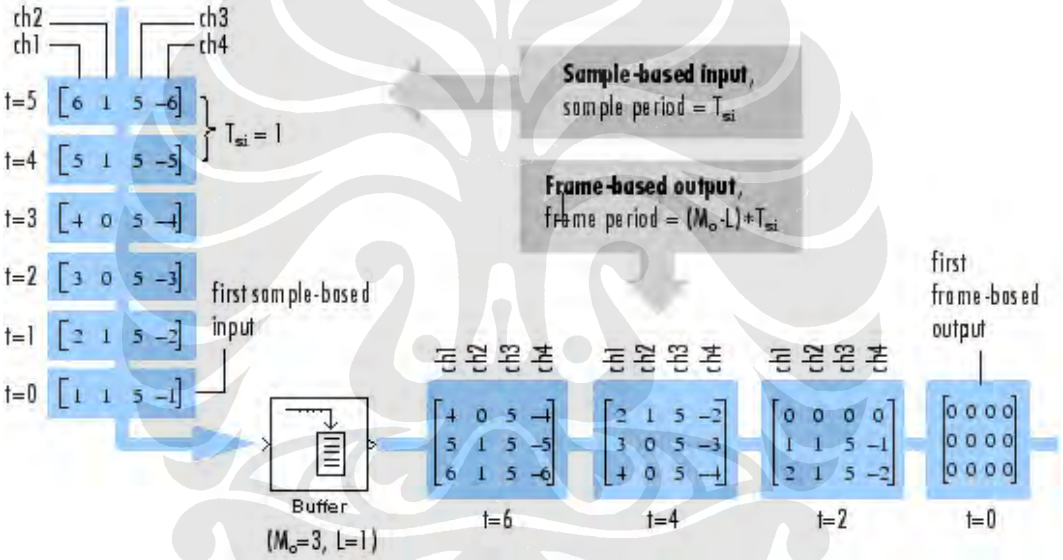


Gambar 3.9. Hasil input video dengan *sample based* [20]

ii. *Buffer*

Blok *buffer* ini berfungsi untuk meredistribusi *sample* input ke sebuah frame baru, dengan ukuran yang lebih besar atau lebih kecil daripada input.

Buffer memperlakukan operasi *sample/frame based* secara berbeda. Pada *sample based*, input sebagai channel data yang independen. Oleh karenanya, *sample* dengan panjang N-vektor akan dianggap sebagai N-channel yang berbeda. Sebuah urutan dari *sample based* dengan panjang input N-vektor (1-D atau 2-D) di-*buffer* menjadi matriks berukuran *Mo* x *N*. Dimana *Mo* dispesifikasi oleh parameter *output buffer size*. Input *Full dimensi* matriks tidak dapat diterima oleh buffer. Gambar 3.11. menunjukkan ilustrasi dari buffer 4-channel *sample based* dengan *output buffer size* 3 dan *buffer overlap* 1.



Gambar 3.10. Prinsip kerja dari blok *buffer* [20]

Konfigurasi yang dilakukan pada simulasi adalah :

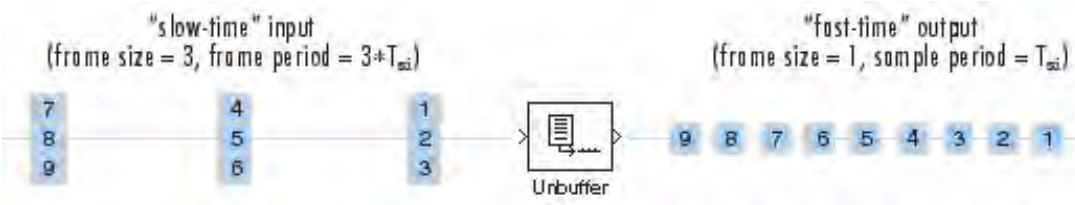
- *Output buffer size* → **64**
- *Buffer overlap* → **0**

Output hasil konfigurasi ini akan menghasilkan matriks berukuran 64 x 120 dalam bentuk sebuah *frame*. Angka 64 dipilih agar sesuai dengan bilangan biner (bit) → 2^6 yang pada nantinya akan dipakai untuk konversi dari bilangan *integer* ke *bit.*

iii. *Unbuffer*

Output dari blok *buffer* akan masuk ke blok *unbuffer*. Blok ini berfungsi untuk memisahkan frame yang berukuran M x N menjadi output 1 x N *sample*

based (matriks baris). Sehingga setiap baris matriks menjadi output independen tiap *sample* waktu. Gambar 3.11. memperlihatkan ilustrasinya.



Gambar 3.11. Prinsip kerja dari blok *unbuffer* [20]

Konfigurasi yang dilakukan adalah :

- *Initial condition* → **0**

Blok *unbuffer* pada simulasi ini berfungsi sebagai pemecah tiap channel. Output matriks pada blok ini akan berukuran 1 x 120. hal ini berarti terdapat 120 channel yang muncul tiap detiknya.

b). Integer to Bit converter

Blok ini berfungsi untuk memetakan setiap input integer (0-255) menjadi sekumpulan grup Bit. Jika M adalah jumlah Bit setiap integer, maka input integer harus terletak antara 0 sampai $2^M - 1$. blok ini akan memetakan setiap integer menjadi sebuah grup yang terdiri dari M Bit dengan menggunakan Bit yang pertama sebagai *most significant bit* (MSB).

Konfigurasi yang dilakukan pada simulasi adalah :

- *Number of bits per integer* → **8**
- *Output data type* → **same as input**

c). Display

Blok ini berfungsi untuk menampilkan semua input yang telah masuk.

Konfigurasi yang dilakukan pada simulasi adalah :

Format → **short**

Decimation → **1**

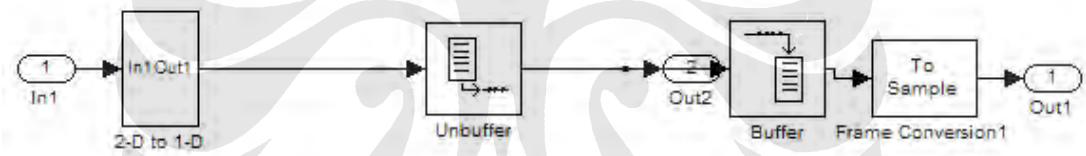
Format *short* artinya blok ini akan menampilkan 5 digit berskala dengan poin desimal tetap. Sedangkan *decimation* dipilih 1 agar menampilkan data setiap tingkat waktu.

d). To workspace

Blok ini berfungsi untuk menuliskan input yang telah masuk ke sebuah *workspace*/ruang kerja. Blok ini menulis outputnya ke sebuah *array* atau struktur yang mempunyai nama yang dispesifikasi dari parameter *variable name*. Tidak ada konfigurasi yang dilakukan pada blok ini.

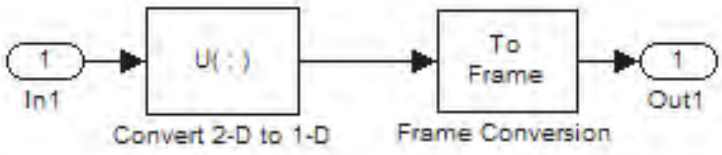
8) Konversi Alat Ukur

Blok ini berfungsi untuk mengkonversi data agar dapat ditampilkan oleh *vector scope.* Seperti yang sudah dijelaskan sebelumnya, bahwa MATLAB 2006b tidak dapat mensimulasikan input data yang besar pada waktu yang bersamaan. Oleh karenanya, data perlu untuk dipaket-paket. Gambar 3.12. memperlihatkan ilustrasinya.



Gambar 3.12. Subsistem Alat ukur 2/3 [18]

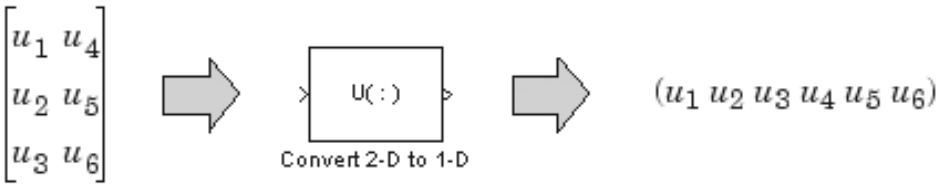
Terlihat pada Gambar 3.12. bahwa blok diagram dari blok *alat ukur 2/3* hampir sama dengan blok *alat ukur 1*, yaitu terdiri dari blok *buffer, unbuffer, frame conversion* dan *2-D to 1-D*. Terdapat 1 blok tambahan pada subsistem *alat ukur 2/3*, yaitu blok *2-D to 1-D*. Gambar 3.13 memperlihatkan ilustrasi dari blok *2-D to 1-D* tersebut.



Gambar 3.13. Subsistem 2-D to 1-D [18]

a) Convert 2-D to 1-D

Blok ini mengatur ulang sebuah input matriks MxN menjadi sebuah vektor dengan panjang MxN. Gambar 3.14. memperlihatkan ilustrasinya.



Gambar 3.14. Prinsip kerja dari blok *convert 2-D to 1-D* [18]

b) Vector scope

Blok ini berfungsi untuk menampilkan data dalam bentuk *time-domain, frequency domain* atau *user-define signal*. Fungsi sebenarnya hampir sama dengan *digital osiloskop*.

9) Integer Delay

Blok ini berfungsi untuk menunda input yang masuk ke blok ini dengan N *sample* periode. Blok ini menerima satu input dan membangkitkan satu output. Data masukkan dapat berbentuk scalar maupun vector.

Konfigurasi yang dilakukan pada simulasi :

- *Sample time* → **1/15, 1/30, 1/60**
- *Number of delays* → **30, 50, 70**

Data masukan ini diambil dari manipulasi *sample time* yang digunakan dan juga tabel yang diambil dari internet.

10) Frame Rate Display

Blok ini berfungsi untuk mengkalkulasikan dan menampilkan jumlah *frame* dari sinyal input dalam *fps* (*frame per second*).

Konfigurasi yang dilakukan pada simulasi :

- *Calculate and display rate every* → 10

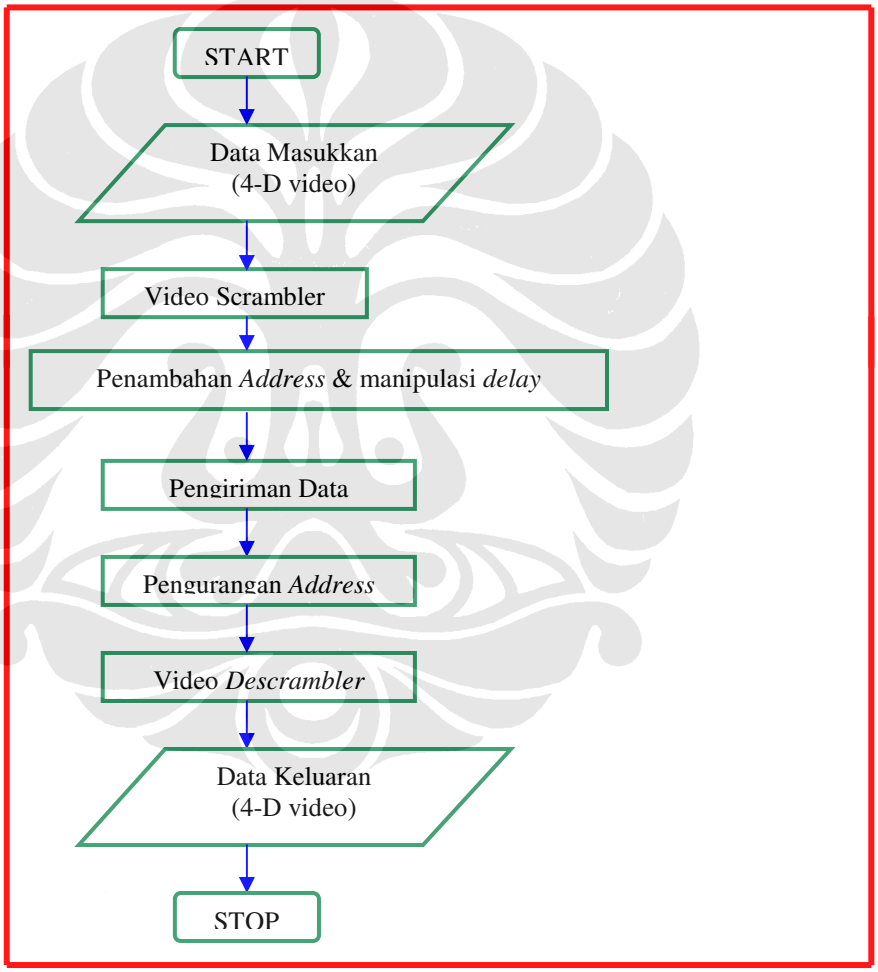
Hal ini berarti blok ini akan menampilkan seluruh *frame* sinyal input tiap interval 10 *frame*.

3.2. SKENARIO SIMULASI VIDEO SCRAMBLER

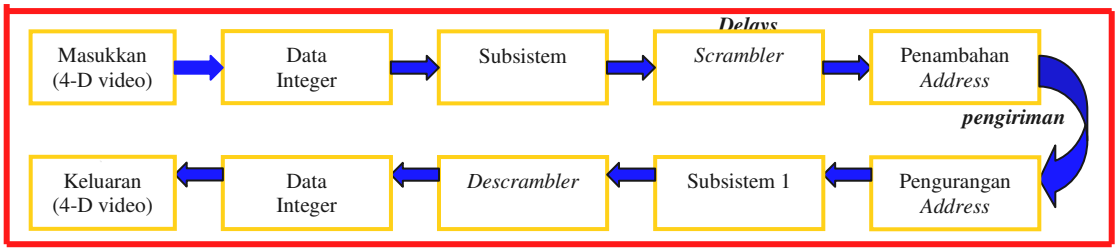
Simulasi dari skripsi ini bertujuan untuk memodelkan video scrambler dengan pengacakan *pixel* disertai dengan penambahan *address code* dengan variasi 2 alternatif, yaitu :

1. Perbandingan *sample time* berbeda (format .**AVI**)
2. Perbandingan *number of delay* untuk tiap *sample time*

Oleh karena itu agar dapat lebih memahami simulasi video *scrambler* tersebut, maka ditampilkan *flow chart* dari simulasi dan penyederhanaan *blok diagram* seperti yang diperlihatkan pada Gambar 3.15. dan Gambar 3.16.



Gambar 3.15. *Flow Chart* simulasi variasi Video *scrambler*



Gambar 3.16. Blok Diagram utama simulasi *scrambling* video

Input dari simulasi video *scrambler* ini berasal dari blok *from multimedia file*, dimana input file name videonya diperoleh dari library MATLAB 2006b yang terletak pada MATLAB → *Toolbox* → *vipblks* → *vipdemos* → *vipmen.avi.* Input video ini seperti yang sudah dijelaskan pada sub-bab sebelumnya merupakan data 4 dimensi (120x160x301) dan satu dimensi lagi untuk perwaktuan.

BAB 4
ANALISIS SISTEM *SCRAMBLING* VIDEO

Simulasi dilakukan untuk mengetahui unjuk kerja hasil rancangan simulasi yang telah dibuat. Sebelum dilakukan simulasi, telah dilakukan pengujian awal dari simulasi termasuk konversi alat ukur yang digunakan. Sehingga sudah dipastikan bahwa simulasi dapat berjalan dengan baik.

Hasil simulasi dilakukan dengan melihat hasil tampilan yang tampak pada 3 blok diagram, yaitu :

a) To video display

b) Vector scope (diberi label '*scramble'*,'pengirim','pelanggan 1')

c) Frame rate display

Dimana penjelasan dari masing-masing blok diagram tersebut telah dijelaskan pada Bab 3.

4.1. PROSEDUR SIMULASI

Simulasi ini dilakukan pada program MATLAB 2006b. Seperti yang sudah dijabarkan pada Bab 3, simulasi dibagi menjadi 3 bagian, yaitu :

1) Simulasi *scrambling* video
2) Variasi *sample time*
3) Variasi *number of delays*

4.1.1. Simulasi *scrambling* video

Simulasi ini dilakukan untuk mengetahui hasil tampilan *scrambling* video dari rancangan yang telah dibuat. Hasil tampilan video nantinya akan ditampilkan pada blok *to video display* dengan 4 kategori, yaitu :

1) *Randomize pixel/* manipulasi pixel
2) *Scrambling*
3) Pengirim
4) Pelanggan

4.1.2. Variasi *Sample time*

Variasi ini dilakukan untuk melihat jumlah *frame per sekon* (*fps*) yang dihasilkan dari blok *Multimedia From File* pada setiap simulasinya. *File Name* video yang diambil beserta Konfigurasi dari blok ini telah dijelaskan pada Bab 3. Kondisi awal/*default sample time* dari video ini diperlihatkan pada Tabel 4.1.

Tabel 4.1. Kondisi *default* video *vipmen.avi*

FILE NAME	***SAMPLE TIME***	**DURATION**	**RESOLUTION**
Vipmen. avi	1/30	10 s	120 X 160

Sumber : www.video and image_processing.org

Terlihat pada Tabel 4.1. bahwa kondisi *default* dari Vipmen.avi berdurasi 10 sekon dengan *sample time* 1/30. Hal ini berarti Vipmen. Avi memiliki 301 fps (1 frame tambahan untuk *blanking interval*).

Pada simulasi skripsi ini, nilai masukkan *sample time* dilakukan dengan 3 alternatif, yaitu :

1) *Sample time* → **1/15** (rendah)
2) *Sample time* → **1/30** (normal)
3) *Sample time* → **1/60** (tinggi)

Output simulasi *sample time* ini dapat dilihat pada blok *vector scope* 3 kategori, yaitu :

1) Pengirim
2) Pelanggan
3) *Scrambling*

4.1.3. Variasi *Number of Delays*

Variasi ini dilakukan untuk mengatur banyaknya penundaan/*delays* yang terjadi pada setiap detiknya. Nilai masukkan untuk *delays* yang dilakukan pada simulasi ini tunjukkan pada Tabel 4.2.

Tabel 4.2. Nilai masukan *delays*

***Jenis* Delays**	**Durasi (per *periode* sample)**
Kecil	**30**
Sedang	**50**
Besar	**70**

Sumber : simulinklibrarybrowser

Output variasi *number of delays* ini akan ditampilkan pada blok-blok diagram yang sama dengan variasi *sample time*.

4.2. ANALISIS HASIL SIMULASI

Setelah melakukan prosedur dengan konfigurasi seperti yang telah dijelaskan pada Bab 3 dan manipulasi data masukkan seperti yang telah dijelaskan pada sub-bab diatas, maka diperoleh beberapa hasil uji coba dan analisis terhadap hasil uji coba yang akan dijabarkan sebagai berikut :

4.2.1. Analisis Video *Scrambler*

Scrambling video dilakukan pada pihak pengirim. Pada simulasi ini, video *scrambler* terdiri dari beberapa proses, yaitu pembagian pixel, *scrambling* dan manipulasi pixel, serta penambahan *address code.*

Seperti yang sudah dijelaskan bahwa output video dari blok *from multimedia file* adalah suatu bentuk matriks yang berukuran 120x160x301, dimana tiap elemen dari matriks tersebut akan mewakili kontras, warna dan detail gambar dari video. Bagian video yang di-*scrambling* adalah bagian R, sedangkan G dan B dikirimkan tanpa adanya manipulasi. Gambar 4.1. akan menampilkan tampilan asli video yang akan dikirimkan ke pihak pelanggan :

Gambar 4.1. Tampilan asli video

4.2.1.1. Pembagian Pixel

Pada pihak pengirim, output matriks R dari blok *multimedia from file* juga akan masuk ke dalam blok *subsystem*. Dalam blok ini, input video yang telah berubah menjadi matriks 2-D akan dibagi-bagi menjadi 6 bagian utama. 6 bagian

utama ini pada nantinya akan menjadi sekumpulan pixel untuk tampilan video. Konfigurasi bagian-bagian tersebut terlihat seperti pada Tabel 3.4.

Tabel 3.4. Pembagian *pixel* video

1	**80**	**160**
	Pixel 1	*Pixel* 4
40	*Pixel* 2	*Pixel* 5
80	*Pixel* 3	*Pixel* 6
120		

Dalam skripsi ini, digunakan 6 pembagian pixel. Akan tetapi pada kenyataannya, pixel video dapat dibagi menjadi n-pixel yang tak terbatas. Terlihat pada Tabel 3.4. diatas bahwa masing-masing pixel akan berukuran sama yaitu, 40x80. Jika output dari masing-masing pixel ini ditampilkan pada blok *display*, maka hasil tampilannya akan tampak seperti pada Gambar 4.2.

Gambar 4.2. Hasil tampilan pembagian Pixel bagian R

4.2.1.2. *Scrambling* dan Manipulasi Pixel

4.2.1.2.1. *Scrambling*

Setelah pixel video terbagi-bagi menjadi 6 pixel utama seperti Gambar 4.2., maka proses selanjutnya yang merupakan langkah terpenting dalam proses video *scrambling* adalah tingkat keamanan dari video *scrambling* itu sendiri. Output dari blok *subsystem* akan masuk ke blok *scrambler*. Akan tetapi, sebelum masuk kedalam blok *scrambler*, terlebih dahulu pixel video diacak urutannya dimana urutan dari pengacakan tersebut hanya diketahui oleh pembuat program. Urutan

dari masing-masing 6 pixel tersebut memiliki 720 kemungkinan (6x5x4x3x2x1). Akan tetapi dalam simulasi ini diambil urutan seperti yang ditunjukkan berikut :

Scrambler

Pixel 1 → Pixel 6

Pixel 2 → Pixel 5

Pixel 3 → Pixel 4

Pixel 4 → Pixel 3

Pixel 5 → Pixel 2

Pixel 6 → Pixel 1

Para pembuat program *scrambling* ini dapat men-*switch* urutan dari masing-masing pixel tersebut sesuai dengan keinginan mereka. Semakin banyak pixel yang terbagi maka semakin banyak kemungkinan urutan pixel video yang diciptakan.

4.2.1.2.2. Manipulasi Pixel

Setelah urutan pixel video diacak, masing-masing pixel tersebut akan ditambah tingkat keamanannya dengan menambah tingkat kesulitannya. Masing-masing pixel akan masuk ke dalam blok diagram yang berisi manipulator-manipulator matriks. Manipulator matriks tersebut dapat berupa :

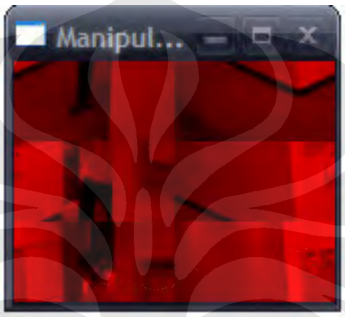
1.*Matrix factorizations*

2.*Matrix inverses*

3.*Matrix operations*

Para pembuat program dapat dengan bebas memanipulasi matriks-matriks tersebut dengan menggunakan manipulator-manipulator matriks. Akan tetapi, pada simulasi ini digunakan blok *matrix operations* → *matrix transpose*. Sehingga 6 pixel yang berukuran 40x80 akan diubah manjadi 80x40. Lalu masing-masing pixel video tersebut digabungkan satu sama lain dengan 2 *matrix concatenation* dengan mode vertikal. Mode vertikal akan menambah jumlah kolom matriks dari masing-masing pixel video. Input dari *matrix concatenation* ada 3 dan output dari *matrix concatenation* ada 1. Sehingga harus dipilih 3 urutan pixel video yang akan digabungkan dimana urutan tersebut harus sama dengan urutan yang telah dibuat diatas.

Output dari *matrix concatenation* adalah matriks yang berukuran 80x120. Agar sama dengan ukuran video aslinya, maka 2 *matrix concatenation* tersebut masing-masing harus di*transpose* lagi (120x80) dan keduanya digabungkan dengan *matrix concatenation* dengan mode yang sama. Output dari *matrix concatenation* yang terakhir adalah matriks yang berukuran sama seperti aslinya, 120x160. Akan tetapi masing-masing pixelnya sudah ditambah dengan manipulasi dari pengacakan pixel. Jika ditampilkan ke *display*, maka tampilannya akan ditunjukkan oleh Gambar 4.3.



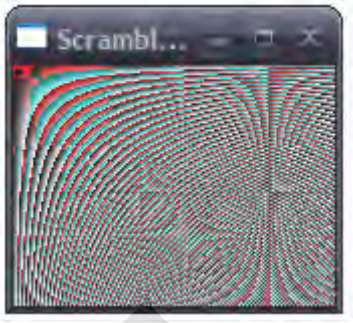
Gambar 4.3. Hasil tampilan pengacakan pixel

4.2.1.3. Penambahan *address code*

Setelah pengacakan pixel berhasil, output dari blok *scrambler* ini akan masuk ke dalam blok *adder*. Seperti yang sudah dijelaskan bahwa blok *adder* adalah blok yang berfungsi untuk menjumlahkan matriks antara matriks hasil pengacakan pixel dan matriks *address code*.

Matriks-matriks hasil dari blok *address code*, dapat berupa pembangkit matriks apapun dan angkanya boleh apa saja tergantung dari pembuat program. Akan tetapi, matriks-matriks yang dibangkitkan tersebut harus sama ukuran pixelnya dengan ukuran dari video aslinya. Hal ini disebabkan karena untuk melakukan aljabar matriks, ukuran dari matriks-matriks yang akan diproses ukurannya harus sama. Dalam simulasi ini digunakan 2 blok *DSP constant*, dimana blok yang pertama terdiri dari 120 urutan angka (matriks baris) dan blok yang kedua terdiri dari 160 urutan angka (matriks kolom). Seperti yang sudah disebutkan bahwa angka tersebut tidaklah harus berurutan. Pembuat program bebas untuk menginisialisasinya sendiri asal ukuran pixelnya sama.

Output dari blok *adder* ini nantinya akan dikirimkan ke pihak pelanggan. Jika hasil ditampilkan pada *display*, maka tampilannya akan terlihat seperti pada Gambar 4.4.



Gambar 4.4. Tampilan hasil pengacakan pixel dan *address code*

4.2.2. Analisis Video *Decrambler*

Pada pihak pelanggan dilakukan proses *descrambling*. Rangkaian proses *descrambling* ini merupakan kebalikan dari proses *scrambling*. Pada simulasi ini, proses *descrambling* terdiri dari beberapa proses, yaitu pembagian pixel dan pengurangan *address code*, *descrambling* manipulasi pixel, serta penggabungan pixel. Berikut adalah rangkaian proses tersebut.

4.2.2.1. Pembagian Pixel dan Pengurangan *address code*

Setelah video asli di*scramble* dengan pengacakan pixel dan *address code*, agar TV pelanggan dapat mengenali video aslinya maka video *scramble* harus dipisahkan dari *adderess code*-nya. Hal ini dapat dilakukan dengan mengurangkan matriks-matriks video *scramble* dengan matriks dari *address code* yang telah dibangkitkan. Hal ini menghasilkan pixel-pixel yang belum terurut. Lalu pixel-pixel tersebut dipecah/dibagi-bagi lagi dengan pembagian yang sama dengan pihak pengirim, yaitu 40x80. Hal ini dengan sukses dilakukan oleh blok *subsystem* 1 dimana konfigurasinya sama dengan blok *subsystem* yang terletak pada pihak pengirim.

4.2.2.2. *Descrambling* dan manipulasi pixel

4.2.2.2.1. *Descrambling*

Agar urutan pixel kembali seperti semula, dekoder pelanggan cukup melakukan re-*ordering* pixel-pixel dimana urutannya harus sama dengan urutan

yang dilakukan pada *scrambler* pengirim. Pada simulasi ini, urutan pixel dai *descrambler* adalah sebagai berikut :

Descrambler

Pixel 6 → Pixel 1

Pixel 5 → Pixel 2

Pixel 4 → Pixel 3

Pixel 3 → Pixel 2

Pixel 2 → Pixel 5

Pixel 1 → Pixel 6

4.2.2.2.2. Manipulasi Pixel

Setelah urutan dari pixel video kembali seperti semula, maka pixel-pixel tersebut perlu digabungkan kembali akan tetapi masih terdapat hasil manipulasi pixel. Agar video hasil manipulasi pixel kembali seperti semula, perlu dilakukan operasi dari blok manipulator-manipulator matriks sekali lagi. Hal ini akan mengakibatkan hilangnya manipulasi yang telah dilakukan pada masing-masing pixel sehingga yang tersisa hanyalah pixel-pixel yang belum tergabung menjadi satu.

4.2.2.3. Penggabungan Pixel

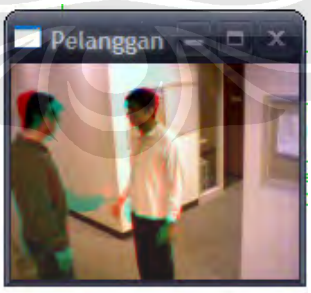
Setelah urutan dari pixel video kembali seperti semula, maka pixel-pixel tersebut digabungkan. Oleh karenanya digunakan 2 blok *matrix concatenation* dimana konfigurasinya sama dengan *matrix concatenation* pada pihak pengirim. 3 kumpulan pixel masing-masing digabungkan dengan *matrix concatenation* dengan mode vertikal sehingga menghasilkan matriks output yang berukuran 80x120. Matriks tersebut di*transpose* sehingga menghasilkan matriks 120x80. masing-masing matriks 120x80 hasil dari *matrix concatenation* digabungkan dengan *matrix concatenation* dengan mode yang sama sehingga menghasilkan matriks output berukuran 120x160 yang sama dengan video aslinya. Jika tampilan pelanggan dimunculkan ke *display*, maka tampilannya akan tampak seperti Gambar 4.5.

Gambar 4.5. Tampilan video pelanggan

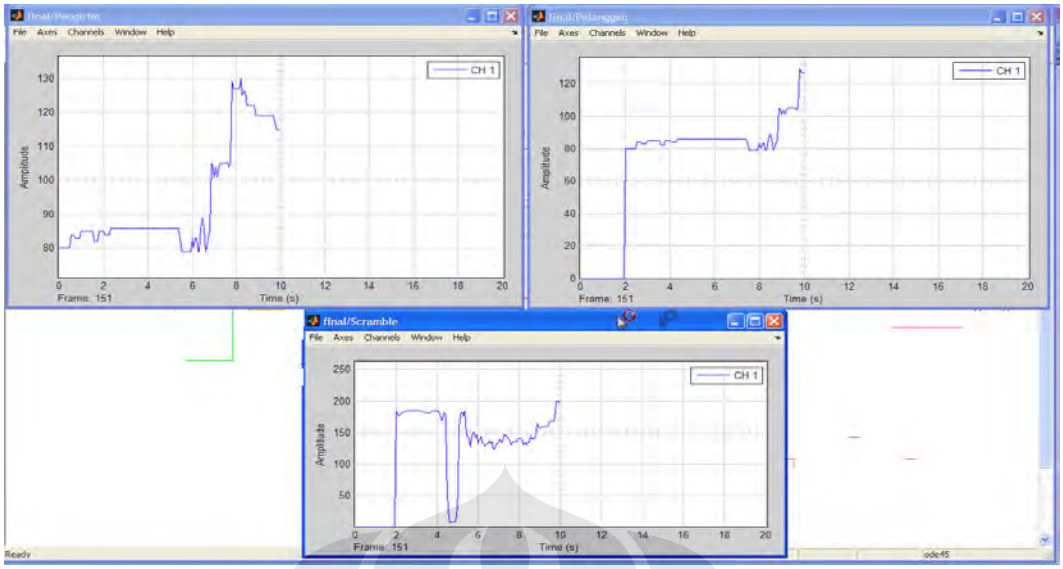
Terlihat bahwa video berhasil direkonstruksi kembali dengan sukses. Perwaktuan yang terjadi antara pihak pengirim dan pihak pihak pelanggan juga sama.akan tetapi pada kenyataannya, setelah dilakukan transmisi pasti ada *delay* akibat pentransmisian tersebut. Oleh karenanya pada hasil analisis simulasi berikut akan dilakukan variasi terhadap video *scrambler* tersebut dengan alternatif terhadap *sample time* dan *delay*.

4.2.2. Analisis *sample time* → 1/15

Setelah dilakukan simulasi dengan konfigurasi dan manipulasi seperti yang telah dijelaskan pada sub-bab diatas, diperoleh hasil simulasi seperti yang tampak pada Gambar 4.6.dan Gambar 4.7.



Gambar 4.6. Tampilan video pelanggan untuk *sample time* 1/15



Gambar 4.7. Bentuk grafik untuk *delay* 30 pada *sample time* 1/15

Bentuk grafik untuk tiap *delay* hampir sama. Hanya saja awal *start* video pada sisi pelanggan saja yang berbeda. Oleh karenanya hanya diambil grafik untuk *delay* 30 sebagai perwakilan. Pada gambar-gambar tersebut, terlihat bahwa durasi video pelanggan menjadi lebih panjang dari seharusnya (10s), yaitu menjadi **30/15 x 10s = 20s**. Hal ini dikarenakan skala ukur yang digunakan memang di-*setting* untuk video berdurasi 10s. Sehingga jika *sample time* lebih rendah dari *default,* maka dapat dilakukan penambahan durasi untuk tampilan video pada pelanggan ataupun pengurangan jumlah *frame* yang di*sample* oleh alat ukur menjadi 151 *frame*.

Pada Gambar 4.6. tampak bahwa simulasi tampilan video terlihat agak patah-patah sehingga tampilan yang muncul untuk pelanggan lebih buruk daripada pengirim. Hal ini terjadi karena terdapat pengurangan jumlah *frame* dari yang seharusnya, yaitu sebesar :

$$300\ frame/\mathbf{2} = \pm 150\ frame$$

Pengurangan jumlah *frame* yang terjadi antara pengirim dan pelanggan, untuk lebih jelasnya diperlihatkan oleh Tabel 4.3.

Tabel 4.3. *Frame rate* untuk tiap *delay*

Sample Time 1/15		
Delay	**Pengirim**	**Pelanggan**
30	29,0695	14,8811
50	29,1547	14,8809
70	26,6665	13,3332

Terlihat bahwa *frame rate* pada pelanggan selalu lebih kecil untuk tiap *delay*. Hal ini selain mengakibatkan tampilan video patah-patah juga kecepatan gerak video menjadi lebih cepat karena *frame rate* pelanggan berusaha menyamai *frame rate* pengirim. Simulasi pada MATLAB 2006b ini dilakukan dengan *Real-time condition* sehingga tampilan video pelanggan seolah-olah saling kejar-mengejar dengan tampilan video pengirim. Selain itu juga, adanya *delay* akan memberikan efek "*ghost*" pada tampilan video pelanggan.

Pada Gambar 4.7. terlihat bahwa masing-masing *delay* akan memberikan perwaktuan efek "*ghost*" yang berbeda-beda pada tampilan video pelanggan. Hal ini dapat dihitung dengan :

Delay 30 → 30/15 = pada detik **ke-2**

Delay 50 → 50/15 = pada detik **ke-3.33**...

Delay 70 → 70/15 = pada detik **ke-4.667**

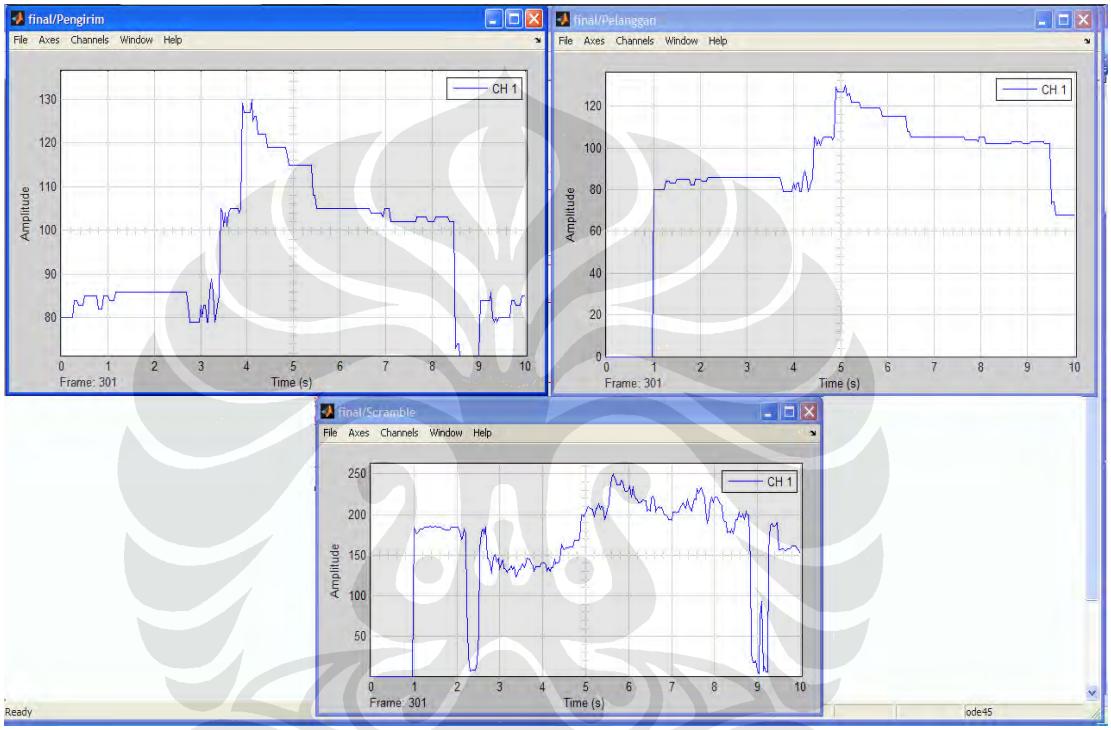
Perbedaan perwaktuan efek "*ghost*" ini akan memperparah tampilan video pelanggan. Terlihat bahwa makin besar *delay*, maka makin jelas pula efek kerusakan yang ditimbulkannya. Efek ini akan semakin terlihat *sample time* yang di-*setting* ternyata lebih rendah daripada *default.*

Pada Gambar 4.7. juga terlihat bahwa untuk *sample time* 1/15, untuk masing-masing *delay* terdapat peningkatan grafik yang curam dan jika diamati lebih detail, amplitudonya sama persis dengan kondisi amplitudo video aslinya. Hal ini akan mengakibatkan ketepatan pada kontras, warna dan detail gambar.

4.2.3. Analisis *sample time* → 1/30

Setelah dilakukan simulasi dengan konfigurasi dan manipulasi seperti yang telah dijelaskan pada sub-bab diatas, diperoleh hasil simulasi seperti yang tampak pada Gambar 4.8. dan Gambar 4.9.

Gambar 4.8. Tampilan video untuk *sample time* 1/30



Gambar 4.9. Bentuk grafik untuk *delay* 30 pada *sample time* 1/30

Terlihat bahwa kondisi ini memang merupakan kondisi *default* dari video. Sehingga tidak terdapat penambahan atau pengurangan durasi video ataupun konversi alat ukur.

Pada Gambar 4.8. tampak bahwa simulasi tampilan video pelanggan terlihat bagus. Hal ini terjadi karena pada *setting* inilah video beroperasi. Tidak ada pengurangan jumlah *frame* dan durasi video. Video yang diterima untuk pelanggan pun sama baiknya dengan video yang dikirimkan. Tabel 4.4. memperlihatkan seragamnya *frame rate* antara pengirim dengan pelanggan.

Tabel 4.4. *Frame rate* untuk tiap *delay*

Sample Time 1/30		
Delay	**Pengirim**	**Pelanggan**
30	22,8312	22,8832
50	18,3149	18,2815
70	15,6008	15,2208

Terlihat bahwa *frame rate* antara pengirim dengan pelanggan hampir sama. Hal ini akan mengakibatkan tampilan video pada pelanggan sama seperti aslinya (tidak patah-patah) dan juga kecepatan gerak video sama persis dengan aslinya. Akan tetapi, efek ”ghost” yang terjadi akibat besarnya *delay* masih terlihat cukup jelas walaupun sudah agak berkurang.

Pada Gambar 4.9. terlihat bahwa masing-masing *delay* akan memberikan perwaktuan efek ”*ghost*” yang berbeda-beda pada tampilan video pelanggan. Hal ini dapat dihitung dengan :

Delay 30 → 30/30 = pada detik ke-**1**

Delay 50 → 50/30 = pada detik ke-**1.667**

Delay 70 → 70/30 = pada detik ke**- 2.333**

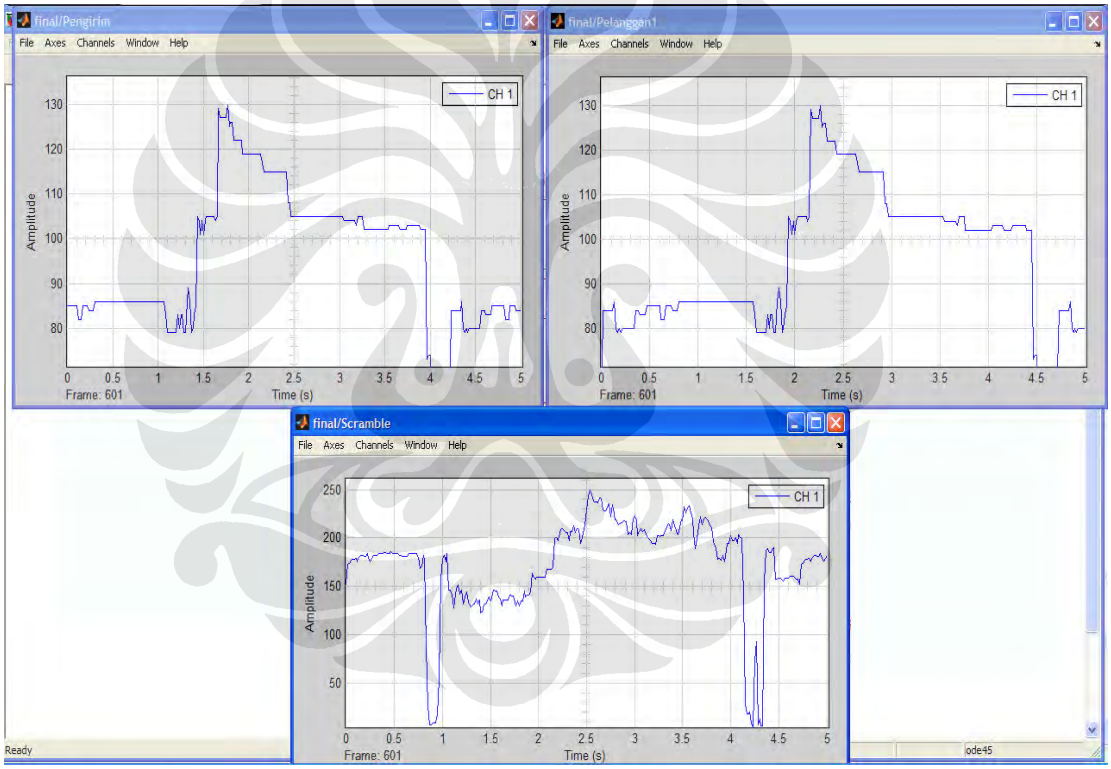
Terlihat bahwa semakin besar *delay,* maka akan semakin besar efek kerusakan yang dihasilkan. Akan tetapi, jika dibandingkan dengan *sample time* → 1/15, maka pada *sample time* ini terdapat perbaikan tampilan video. Hal ini terlihat dari semakin kecilnya nilai *start* dari efek tersebut. Dengan semakin mengecilnya nilai *start* dari efek tersebut, maka tampilan video pelanggan akan hampir sama dengan pengirim.

Selain itu, terdapat ketepatan pada kontras, warna dan detail gambar. Hal ini dapat dilihat pada Gambar 4.9. Masing-masing *delay* memperlihatkan kenaikan grafik yang curam (awal *start*) pada pihak pelanggan, konstannya kenaikan amplitudo yang terjadi, serta seragamnya durasi antara pihak pengirim dan pihak penerima.

4.2.4. Analisis *Sample Time* → 1/60

Setelah dilakukan simulasi dengan konfigurasi dan manipulasi seperti yang telah dijelaskan pada sub-bab diatas, diperoleh hasil simulasi seperti yang tampak pada Gambar 4.10. hingga 4.11.

Gambar 4.10. Tampilan video untuk *sample time* 1/60



Gambar 4.11. Bentuk grafik untuk *delay* 30 pada *sample time* 1/60

Bentuk grafik untuk tiap *delay* hampir sama. Hanya saja awal *start* video berbeda. Sehingga digunakan *delay* kecil sebagai perwakilan. Pada gambar-gambar tersebut terlihat bahwa durasi video menjadi lebih cepat dari kondisi *default* (10s), yaitu menjadi **30/60 x 10s = 5s**. Hal ini dikarenakan *setting* alat ukur memang digunakan untuk video kondisi *default*. Sehinga perlu dilakukan suatu perubahan atau pengaturan lagi. Pengaturan dilakukan untuk durasi tampilan video. Durasinya harus menjadi lebih pendek. Selain itu, harus dilakukan

penambahan jumlah *frame* yang di-*sample* oleh alat ukur menjadi 601 *frame* agar semua input video dapat ditampilkan dengan sempurna.

Pada Gambar 4.10. tampak bahwa simulasi tampilan video disisi pelanggan terlihat mulus dan sangat lambat. Hal ini dapat terjadi karena terdapat penambahan jumlah *frame* dari yang seharusnya, yaitu menjadi :

300 *frame* x **2** = ±600 *frame*

Penambahan *frame rate* antara pengirim dengan pelanggan lebih jelasnya diperlihatkan pada Tabel 4.5.

Tabel 4.5. *Frame rate* untuk tiap *delay*

Sample Time 1/60		
Delay	**Pengirim**	**Pelanggan**
30	12,8042	25,641
50	9,8424	19,3798
70	10,0001	20,0001

Terlihat bahwa *frame rate* pelanggan selalu lebih besar daripada pengirim untuk tiap *delay*. Hal ini akan mengakibatkan tampilan video yang tampak pada pelanggan menjadi sangat lambat. Sisi pelanggan harus menunggu inputan dari *source* video yang memiliki *sample time* yang lebih rendah. Masih terlihat efek ”*ghost*”, akan tetapi cuma sedikit.

Pada Gambar 4.11. terlihat bahwa masing-masing *delay* akan memberikan perwaktuan efek ”*ghost*” yang berbeda-beda pada tampilan video pelanggan. Hal ini dapat dihitung dengan :

Delay 30 → 30/60 = pada detik ke-**0.5**

Delay 50 → 50/60 = pada detik ke-**0.8333**...

Delay 70 → 70/60 = pada detik ke-**1.1667**

Terlihat bahwa semakin besar delay, maka akan semakin besar efek kerusakan yang dihasilkan. Akan tetapi, jika dibandingkan dengan *sample time* 1/15 dan 1/30, kualitas video, dalam hal ini adalah waktu *start* dari efek “*ghost*”, akan menjadi lebih kecil. Terlihat bahwa untuk *delay* yang besar sekalipun, *sample time* 1/60 mampu untuk meminimalisir efek tersebut. Sehingga tampilan video pelanggan akan hampir sama dengan aslinya walaupun gerak video berlangsung lambat.

Pada *sample time* ini, tampilan video pelanggan masih akan memberikan kualitas kontras, warna dan detail gambar yang baik. Hal ini dapat dilihat dari responsifnya/spontannya kenaikan grafik pada pelanggan.

BAB 5
KESIMPULAN

1. Proses *scrambling* melibatkan 2 blok utama, yaitu *subsystem* dan *scrambler*. Pada blok *subsystem* tiap pixel video akan dibagi-bagi matriksnya. Pada blok *scrambler* tiap pixel akan diacak urutannya dan tiap elemen matriksnya akan dimanipulasi sesuai dengan keinginan pembuat program.
2. Tingginya tingkat keamanan rancangan *scrambling* video ini ditentukan dari banyaknya pixel video yang terbagi, penggunaan manipulator-manipulator matriks dan acaknya nilai pembangkitan matriks oleh *address code.*
3. Banyaknya pixel video yang terbagi, maka akan menambah banyaknya kemungkinan urutan pixel video yang dapat dibuat. Dan semakin banyak penggunaan manipulator-manipulator matriks akan menambah besarnya efek kerusakan gambar pada urutan pixel video yang telah dibuat.
4. Rancangan sistem *scrambling* video ini dapat diimplementasikan dengan salah satu metode *scrambling*, yaitu *horizontal sync suppression*. Metode ini akan memisahkan sinyal sinkronisasi (untuk dilemahkan) dari sinyal video (untuk dimanipulasi dengan pengacakan pixel dan *address code*).
5. *Delay* diberikan untuk memberikan variasi perwaktuan dari efek "*ghost*" yang muncul pada tampilan video pelanggan. Semakin besar *delay*, maka akan semakin besar kerusakannya.
6. *Sample time* 1/15 gambar menjadi patah-patah dan kecepatan gerak video menjadi lebih cepat. Efek "ghost" terlihat sangat jelas untuk tiap *delay.* Pada *Sample time* 1/30 durasi video sama dengan aslinya. Akan tetapi efek "ghost" yang terjadi untuk *delay* yang besar masih sangat terlihat. Pada *sample time* 1/60. Tampilan video terlihat mulus tapi gerakannya sangat lambat. Efek "ghost" dapat diminimalisir meskipun untuk *delay* yang besar.

DAFTAR ACUAN

[1] Muhammad Asvial. *Modulation Technique.* (Jakarta : Electrical Engineering Department, University of Indonesia, 2006), hal. 114.

[2] _____. "*History Of Cable Television*". Diakses 16 April 2008 dari NCTA. http://www.ncta.com/About/About/HistoryofCableTelevision.aspx

[3] _____. "*Troubleshooting Cable Issues*". Diakses 12 April 2008 dari time warner cable. http://www.timewarnercable.com/CustomerService/FAQ/TWCFAQCategories.ashx?MarketID=22

[4] _____. "*BASIC HDTV CONNECTIONS*". Diakses 15 April dari time warner cable. http://www.timewarnercable.com/kansascity/customer/selfhelp/default.html

[5] Mistry, Kantilal (Freehold, NJ)." *Method of and apparatus for scrambling and decoding television and similar signals with wobbulating trapping*". Diakses 12 April 2008 dari freepatentsonline. http://www.freepatentsonline.com/4575753.html

[6] Hayashi, Michael Tomoyuki (1999). "*Video Inversion Detection apparatus and Method*". Diakses 16 April 2008 dari freepatentsonline. http://www.freepatentsonline.com/5930361.html

[7] Marland, Dale W (2005). "*Sync suppression television security sistem with addressable sync restoration*". Diakses 20 April 2008 dari freepatentsonline. http://www.freepatentsonline.com/6292567.html

[8] ___. "*Method and Apparatus for Scrambling and Unscrambling Using Encryption and Decryption*". Diakses 15 April 2008. http://www.freepatentsonline.com/4535355.html

[9] Naohiza Kitazato (1997). "*Apparatus for Scrambling a Digital Video Signal*". Diakses 15 April 2008 dari Patent Storm. http://www.patentstorm.us/patents/5600721.html

[10]_____. "*Basic TV Hookup without a Cable Converter*". Diakses 15 April 2008. http://www.timewarnercable.com/CustomerService/FAQ/TWCFAQCategories.ashx

[11] Atkin, Laurence Walter (2000). "*Video Scrambling Sistems*". Diakses 15 April 2008 dari freepatentsonline. http://www.freepatentsonline.com/EP0356200A1.html

[12] _____. "*Cable TV Information*". Diakses 15 April 2008 dari Telecommunication services. http://www.telecom.tcu.edu/st_cable.asp#channel

[13] Gale, Brent. *Scrambling and descrambling satellite and cable TV*. 2000. ISBN : 0-917893-07-07.

[14] _____. " *Method and Apparatus for Performing Frequency Spectrum Inversion*". Diakses 15 April 2008 dari Patent Storm. http://www.patentstorm.us/patents/5796838.html

[15] Kanginan, Marthen. (2002). "*Seribu Pena Fisika Untuk SMU*". Jakarta : Erlangga.

[16] Tampomas, Husein. (2002). "*Seribu Pena Matematika Untuk SMU*". Jakarta : Erlangga.

[17] Tim guru BTA. (2004). "*Teori dan Soal Matematika*". Jakarta " TIM guru BTA 8 Jakarta Selatan.

[18] _____,"*MATLAB Help File*", version 7.0. 3. 267 2006b". TheMathWorks Inc, 2006.

[19] _____,"*video and image_processing simulation*". Diakses 10 Juni 2008 dari Department Information and Science. http://www.video and image_processing.org

[20] _____."*MATLAB Simulink Library Browser*". The mathwork, Inc, 2006.

DAFTAR PUSTAKA

_____. “Cable Installation Instructions”. Diakses 15 April 2008 dari Departement of Housing, University New Hampshire. http://www.unh.edu/housing/catvision/setupguide.html

Naohiza Kitazato (1997). “Apparatus for Scrambling a Digital Video Signal”. Diakses 15 April 2008 dari Patent Storm. http://www.patentstorm.us/patents/5600721.html

_____. “Kamus Besar Bahasa Indonesia”. 2000. Jakarta : Erlangga.