



Syahrul Imardi, MT

#19

MATAKULIAH
KEAMANAN PERANGKAT LUNAK

Steganograf (Bagian 2)



MATAKULIAH KEAMANAN KAT LUNAK

Syahrul Imardi, MT

BAGIAN 2

P19 : Steganograf (Bagian 2)



P19



STMIK
Amik Riau



Steganograf

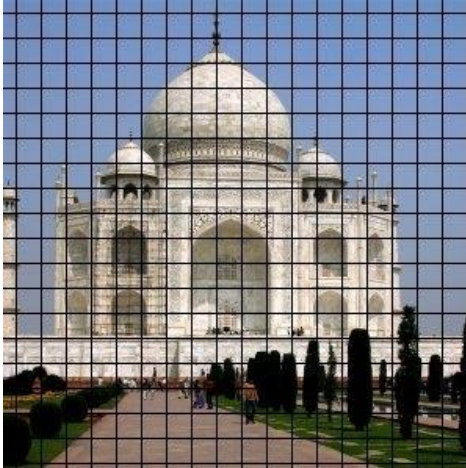
(Bagian 2)

Ranah Steganografi

Berdasarkan ranah operasinya, metode steganografi dapat dibagi menjadi dua kelas:

- *Spatial (time) domain methods*

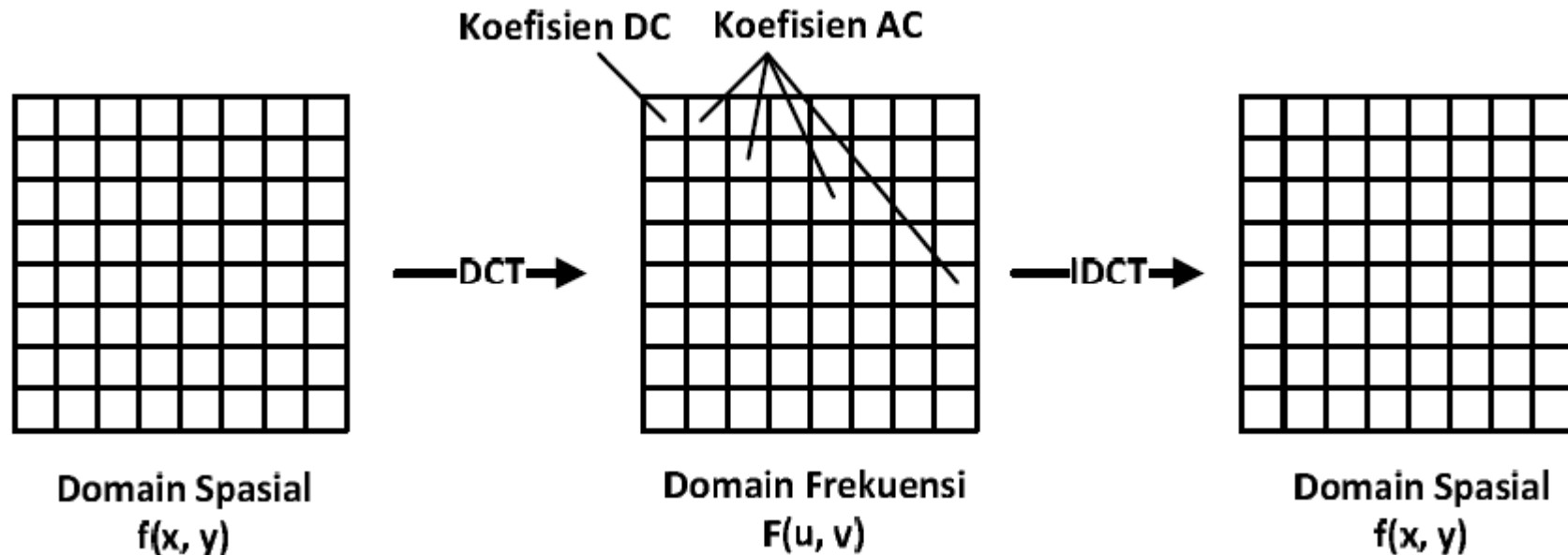
Memodifikasi langsung nilai *byte* dari *cover-object* (*byte* merepresentasikan nilai *pixel* pada data citra atau nilai amplitudo pada data audio)



Contoh: Metode *LSB*, BPCS, echo-hiding, parity-bit coding, dll

- *Tranform domain methods*

Memodifikasi hasil transformasi sinyal dalam ranah *transform* (hasil transformasi dari ranah spasial atau waktu ke ranah lain misalnya ranah frekuensi). Metode transformasi yang digunakan misalnya DCT, FFT, DWT.

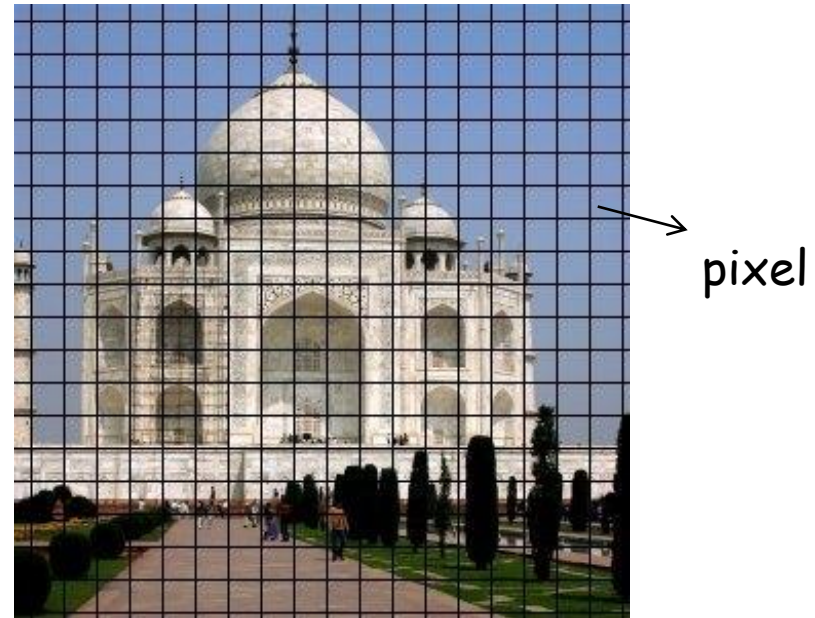


Contoh: Metode *Spread Spectrum*

Metode LSB

Citra Digital

- Citra terdiri dari sejumlah *pixel*. Citra 1200 x 1500 berarti memiliki 1200 x 1500 pixel = 1.800.000 pixel



- Setiap *pixel* panjangnya n -bit.
Citra biner \rightarrow 1 bit/pixel
Citra *grayscale* \rightarrow 8 bit/pixel
Citra *true color* \rightarrow 24 bit/pixel

Citra Lenna



True color image
(24-bit)

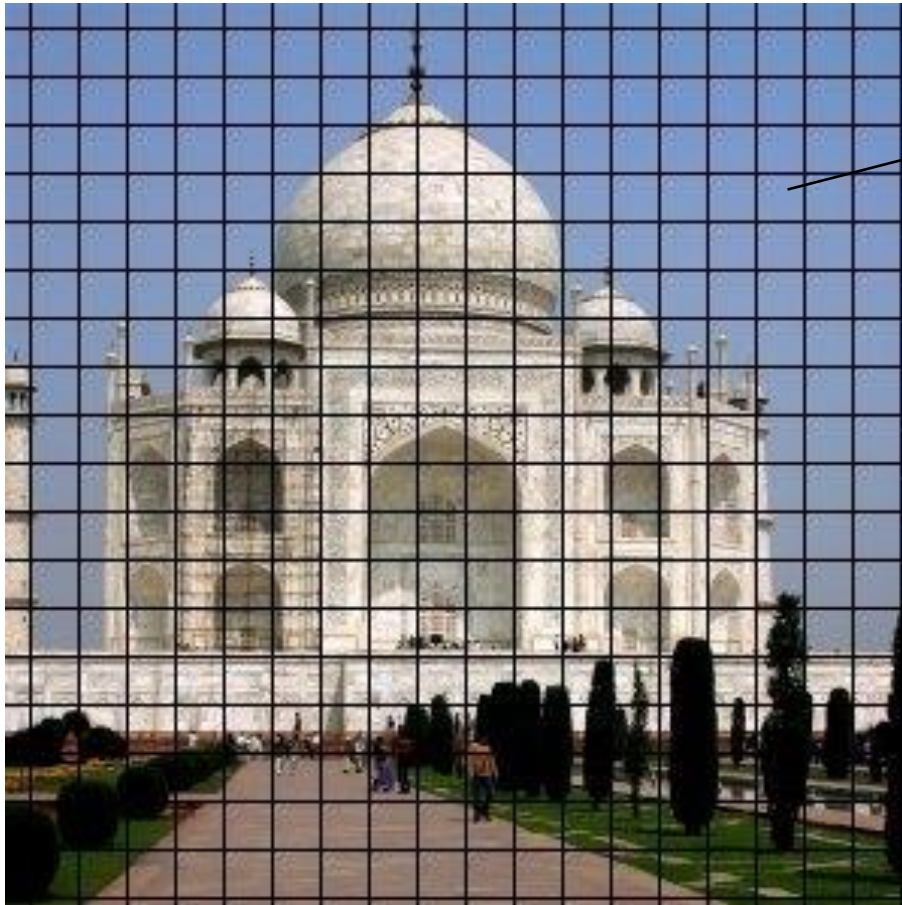


Grayscale image
(8-bit)



Bimary image
(1-bit)

Pada citra 24-bit (*real image*), 1 pixel = 24 bit,
terdiri dari komponen RGB (Red-Green-Blue)



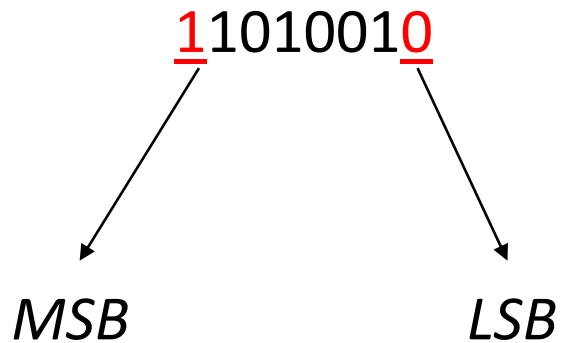
100100111001010010001010
R G B

Bitplane pada Citra Digital

- Nilai *pixel* pada koordinat (x, y) menyatakan intensitas nilai keabuan pada posisi tersebut.
- Pada citra *grayscale* nilai keabuan itu dinyatakan dalam *integer* berukuran 1 *byte* sehingga rentang nilainya antara 0 sampai 255.
- Pada citra berwarna 24-bit setiap *pixel* terdiri atas kanal *red*, *green*, dan *blue* (*RGB*) sehingga setiap *pixel* berukuran 3 *byte* (24 bit).

- Di dalam setiap *byte* bit-bitnya tersusun dari kanan ke kiri dalam urutan yang paling berarti (*most significant bits* atau *MSB*) hingga bit-bit yang kurang berarti (*least significant bits* atau *LSB*).
- Susunan bit pada setiap *byte* adalah $b_7b_6b_5b_4b_3b_2b_1b_0$

Contoh:



LSB = Least Significant Bit
MSB = Most Significant Bit

- Jika setiap bit ke- i dari *MSB* ke *LSB* pada setiap *pixel* diekstrak dan diplot ke dalam setiap *bitplane image* maka diperoleh delapan buah citra biner.



Original image



Bitplane 7



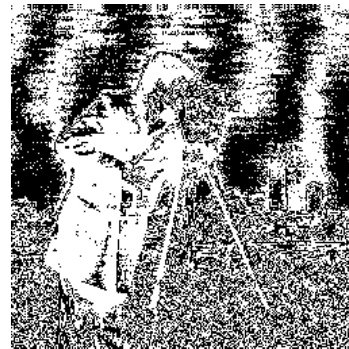
Bitplane 6



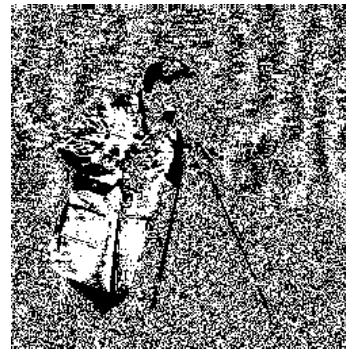
Bitplane 5



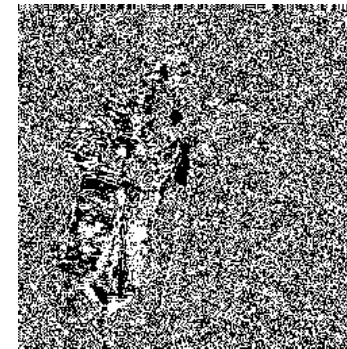
Bitplane 4



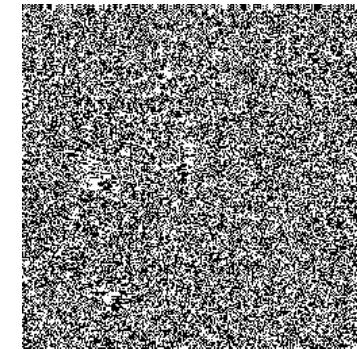
Bitplane 3



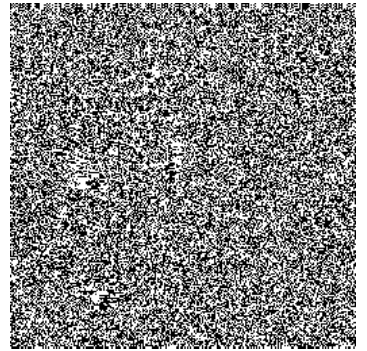
Bitplane 2



Bitplane 1



Bitplane 0

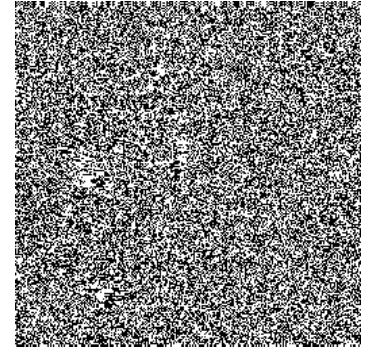


Bitplane 0

- *Bitplane* LSB, yaitu *bitplane* 0, terlihat seperti citra acak (*random image*).
- Bitplane LSB merupakan bagian yang redundan pada citra.
- Artinya, perubahan nilai bit pada bagian tersebut tidak mengubah persepsi citra secara keseluruhan.
- Inilah yang mendasari metode steganografi yang paling sederhana, yaitu **metode LSB**.

Metode LSB

- Merupakan metode steganografi yang paling populer.
- Memanfaatkan kelemahan indra visual manusia dalam mengamati perubahan sedikit pada gambar
- Caranya: Mengganti bit *LSB* dari *pixel* dengan bit pesan.



Bitplane 0

Mengubah bit *LSB* hanya mengubah nilai *byte* satu lebih tinggi atau satu lebih rendah dari nilai sebelumnya → tidak berpengaruh terhadap persepsi visual/auditori.

Misalkan semua bit LSB pada citra berwarna dibalikkan dari semula 0 menjadi 1; dari semula 1 menjadi 0



Sebelum



Sesudah

Adakah terlihat perbedaanya?

Misalkan semua bit LSB pada citra *grayscale* dibalikkan
Dari semula 0 menjadi 1; dari semula 1 menjadi 0



Sebelum



Sesudah

Adakah terlihat perbedaanya?

Contoh 1:

- Tinjau 1 buah *pixel* dari citra 24-bit (3 x 8 bit):

10000010

(130)

01111011

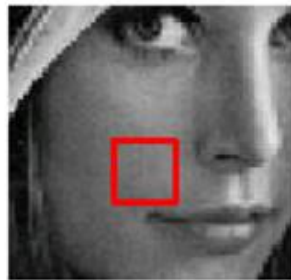
(123)

01110101

(117)

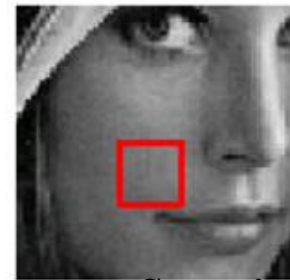
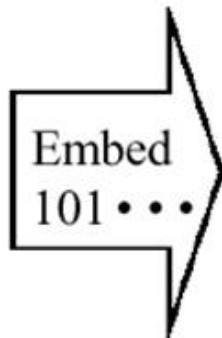
- Bit bit-bit *embedded message*: 101

- *Embed*: 00110011 10100010 11100011



original

130 = 10000010
123 = 01111011
117 = 01110101
.....



Stego-image

131 = 10000011
122 = 01111010
117 = 01110101
.....

Original: misalkan *pixel* [130, 123, 117] berwarna "ungu"
Stego-image: *pixel* [131, 122, 117] tetap "ungu" tapi berubah sangat sedikit.
Mata manusia tidak dapat membedakan perubahan warna yang sangat kecil.

Pergeseran warna sebesar 1 dari 256 warna tidak dapat dilihat oleh manusia

PESAN RAHASIA :



Sumber: TA Yulie Anneria Sinaga 13504085

Contoh 2:

- Jika pesan = 10 bit, maka jumlah *byte* yang digunakan = 10 *byte*

- Contoh susunan *byte* yang lebih panjang:

00110011 10100010 11100010 10101011 00100110
10010110 11001001 11111001 10001000 10100011

- Pesan: 1110010111

- Hasil penyisipan pada bit *LSB*:

0011001**1** 1010001**1** 1110001**1** 1010101**0** 0010011**0**
1001011**1** 1100100**0** 1111100**1** 1000100**1** 1010001**1**

Ekstraksi Pesan dari *Stego-image*

- Bit-bit pesan yang disembunyikan di dalam citra harus dapat diekstraksi kembali.
- Caranya adalah dengan membaca *byte-byte* di dalam citra, mengambil bit LSB-nya, dan merangkainya kembali menjadi bit-bit pesan.
- Contoh: Misalkan *stego-object* adalah sbb

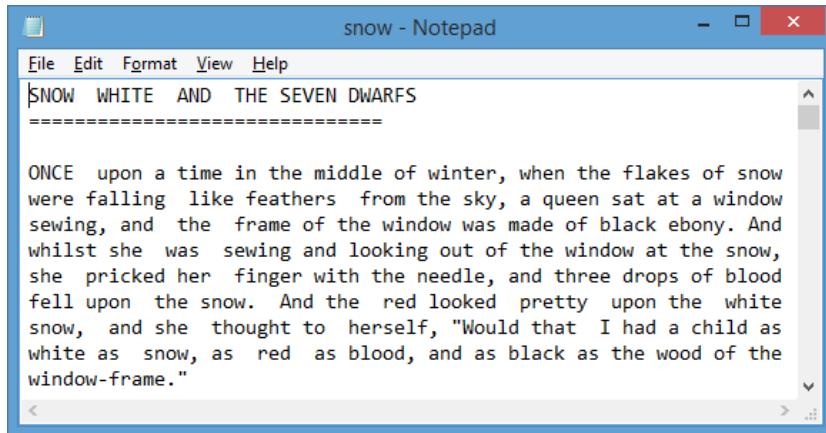
00110011 10100011 11100011 10101010 00100110
10010111 11001000 11111001 10001001 10100011

Ekstrak bit-bit LSB: 1110010111

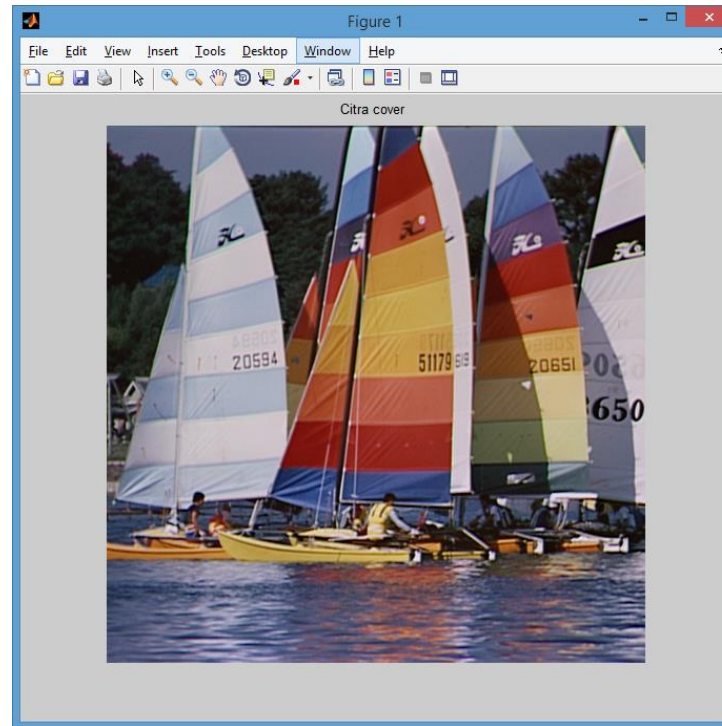
Menghitung Ukuran Pesan yang dapat Disembunyikan

- Ukuran pesan yang akan disembunyikan bergantung pada ukuran *cover-object*.
- Misalkan pada citra *grayscale* (1 *byte/pixel*) 256 x 256 *pixel* :
 - jumlah *pixel* = jumlah *byte* = $256 \times 256 = 65536$
 - setiap *byte* dapat menyembunyikan 1 bit pesan di LSB-nya
 - jadi ukuran maksimal pesan = 65536 bit = 8192 *byte* = 8 KB
- Pada citra berwarna 24-bit berukuran 256×256 *pixel*:
 - jumlah *pixel* $256 \times 256 = 65536$
 - setiap *pixel* = 3 *byte*, berarti ada $65536 \times 3 = 196608$ *byte*.
 - setiap *byte* dapat menyembunyikan 1 bit pesan
 - jadi ukuran maksimal pesan = 196608 bit = 24576 *byte* = 24KB

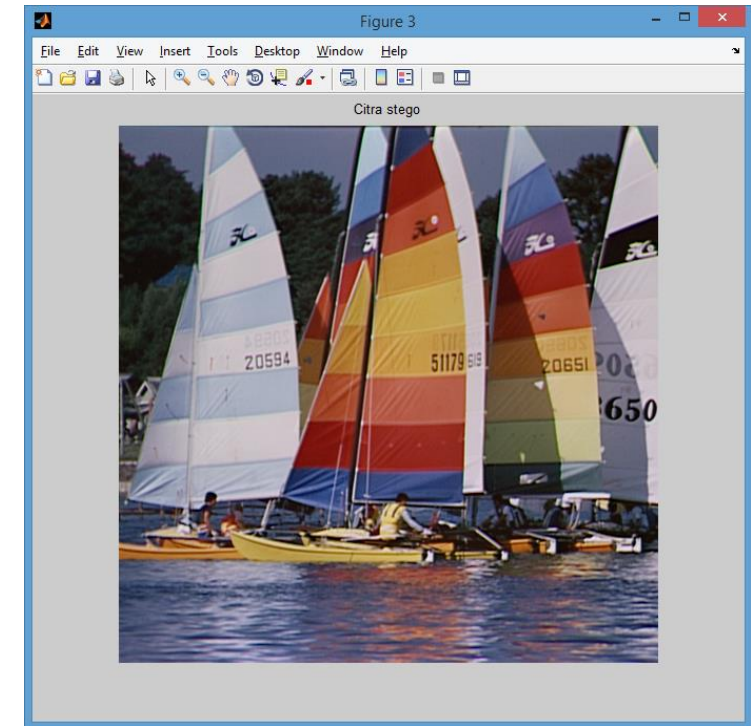
- Contoh penyembunyian pesan teks ke dalam citra



Secret message



Cover-image



Stego-image

- Contoh penyembunyian gambar ke dalam gambar



Secret message
60 KB



Cover-image
769 KB



Stego-image
769 KB



Extracted message
60 KB

Beberapa Varian Metode LSB

1. *Sequential*

- Bit-bit pesan disembunyikan secara sekuensial pada *pixel-pixel* citra.
- Misalkan ukuran pesan = 15 bit, maka urutan *pixel-pixel* yang digunakan untuk penyembunyian bit adalah:

1	2	3	4	5	6	7	8
9	10	11	12	13	14	15	
						=	

Ekstraksi pesan dari *Stego-image*

- Pada proses ekstraksi pesan, *pixel-pixel* dibaca secara sekuensial mulai dari *pixel* pertama sampai *pixel* yang menyimpan bit pesan terakhir
- Ambil setiap *byte* dari *pixel*, ekstraksi bit LSB-nya.
- Rangkailah bit-bit LSB menjadi bit-bit pesan semula.

2. Acak

- Untuk membuat penyembunyian pesan lebih aman, bit-bit pesan tidak disimpan pada pixel-pixel yang berurutan, namun dipilih secara acak.
- Pembangkit bilangan acak-semu (*PRNG: pseudo-random number generator*) digunakan untuk membangkitkan bilangan acak.
- Umpan (*seed*) untuk pembangkit bilangan acak berlaku sebagai kunci (*stego-key*).

- Misalnya jika terdapat 64 *byte* dan 15 bit pesan yang akan disembunyikan. *Pixel-pixel* dipilih secara acak, seperti pada gambar berikut.

				5			8
	10					4	
			13		2		
7							9
		1			12		
		15					
11						3	
			6				14

Ekstraksi pesan dari *Stego-image*

- Posisi *pixel* yang menyimpan bit pesan dapat diketahui dari bilangan acak yang dibangkitkan oleh *PRNG*.
- Jika kunci yang digunakan pada waktu ekstraksi sama dengan kunci pada waktu penyisipan, maka bilangan acak yang dibangkitkan juga sama.
- Dengan demikian, bit-bit pesan yang bertaburan di dalam citra dapat dikumpulkan kembali.

3. *m*-bit LSB

- Untuk meningkatkan ukuran pesan yang disembunyikan, maka digunakan lebih dari 1 bit LSB untuk setiap *byte*.
- Susunan bit pada setiap *byte* adalah $b_7b_6b_5b_4b_3b_2b_1b_0$. Jika diambil 2-bit LSB, maka bit yang digunakan adalah bit b_1 dan bit b_0

Contoh: 11010010 → 2 bit LSB terakhir dipakai untuk menyembunyikan pesan.

- *Trade-off*: Semakin banyak bit LSB yang digunakan, semakin besar ukuran pesan yang dapat disembunyikan, tetapi semakin turun kualitas *stego-image*.
- Pesan dapat disembunyikan secara sekuensial atau secara acak pada *pixel-pixel* di dalam citra.

4. Enkripsi

- Pesan dapat dienkripsi terlebih dahulu sebelum disembunyikan ke dalam citra.
- Teknik enkripsi yang sederhana misalnya dengan meng-XOR-kan bit-bit pesan dengan bit-bit kunci. Jumlah bit-bit kunci sama dengan jumlah bit pesan.
- Bit-bit kunci dibangkitkan secara acak.
- Kunci untuk pembangkitan bit-bit kunci menjadi *stego-key*.
- Jika dipakai teknik acak dalam memilih *pixel-pixel*, maka ada dua *stego-key*: satu untuk pembangkitan bit-bit kunci, satu lagi untuk pembangkitan posisi *pixel* yang dipilih untuk menyembunyikan pesan.

PSNR

- PSNR = *Peak-Signal-to-Noise Ratio*
- Merupakan metrik untuk mengukur kualitas (*fidelity*) citra setelah proses manipulasi.
- Selalu dibandingkan dengan citra semula (yang belum dimanipulasi).
- Misalkan I = *cover-image* dan \hat{I} = *stego-image*, ukuran citra $M \times N$, maka

$$PSNR = 20 \times \log_{10} \left(\frac{255}{rms} \right) \quad \text{dengan} \quad rms = \sqrt{\frac{1}{MN} \sum_{i=1}^N \sum_{j=1}^M (I_{ij} - \hat{I}_{ij})^2}$$

rms = *root mean square*

$$PSNR = 20 \times \log_{10} \left(\frac{255}{rms} \right)$$

- Satuan *PSNR* adalah desibel (dB).
- Nilai *PSNR* berbanding terbalik dengan *rms*.
- *PSNR* yang besar mengindikasikan nilai *rms* yang kecil; *rms* kecil berarti dua buah citra mempunyai hanya sedikit perbedaan.
- *PSNR* yang kecil mengindikasikan nilai *rms* yang besar; *rms* besar berarti kedua citra memiliki perbedaan yang besar (degradasi).
- *PSNR* yang dapat diterima/ditoleransi adalah jika > 30

How to Hack a Computer Using Just An Image

Monday, June 01, 2015 Swati Khandelwal

512 Like 8.5K Share 12.8K Tweet 923 Share 84 share 19.2K

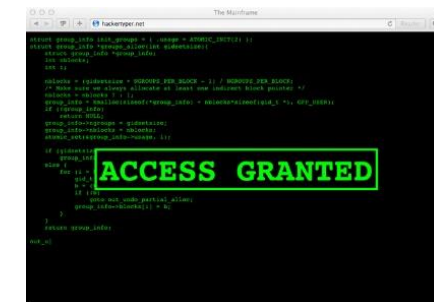


Next time when someone sends you a photo of a cute cat or a hot chick than be careful before you [CLICK](#) on the image to view — it might hack your machine.

Yes, the normal looking images could hack your computers — thanks to a technique discovered by security researcher *Saumil Shah* from India.

Dubbed "*Stegosplit*," the technique lets hackers hide malicious code inside the pixels of an image, hiding a malware exploit in plain sight to infect target victims.

Just look at the image and you are HACKED!



Program Stegano *shareware*

1. InPlainView:

<http://www.simtel.net/product.php%5Bid%5D12796%5BSiteID%5Dsimtel.net>

Keterangan: hanya untuk citra .bmp

2. S-tools

<http://digitalforensics.champlain.edu/download/s-tools4.zip>

Keterangan: untuk citra GIF dan BMP.

- Daftar 100 kakas steganografi lainnya:
<http://www.jjtc.com/Steganography/toolmatrix.htm>
- Beberapa diantaranya berjalan di Linux:
 1. **JPHS (JPHide JPSeek, JP hide and seek)**
<http://linux01.gwdg.de/~alatham/stego.html>
 2. Steghide
 3. Outguess
 4. Blindside
 5. Gifshuffle
 6. GzSteg
 7. dll

- Situs populer untuk informasi steganografi
 - <http://www.ise.gmu.edu/~njohnson/Steganography>
 - <http://www.rhetoric.umn.edu/Rhetoric/misc/dfrank/stegsoft.html>
 - <http://www.topology.org/crypto.html>
 - <http://mozaiq.org/>

BERSAMBUNG
ke Bagian 3

Referensi utama :

Li, F., *The art and science of writing hidden messages: Steganography*

Khan, M. M. , *Steganography*

Wohlgemuth, S. (2002), IT-Security: Theory and Practice : *Steganography and Watermarking*, University of Freiburg, Denmark, 2002.

Wong, P.W. (1997). *A Watermark for Image Integrity and Ownership Verification*.
Prosiding IS&T PIC Conference.

Tawalbeh, L. (2006), Watermarking, Information System Security AABFS-Jordan.

Bae, S.H. (2006), *Copyright Protection of Digital Image*, Tongmyong University of
information technology

Yuli Anneria Sinaga, *Steganalisis dengan Metode Chi-square dan RS-analysis*, Tugas
Akhir Informatika, IT



Ada pertanyaan?

