Criptografía y esteganografía
Una breve experiencia

Eduardo Ruiz Duarte

Facultad de Ciencias UNAM

Abril 23, 2009

Características

- **Criptografía:** Un mensaje cifrado es detectable fácilmente por pruebas probabilísticas
- **Esteganografía:** Un mensaje oculto en la naturaleza de otro no es fácilmente detectable

Características preferentes

Criptografía:

- **No lineal**
  - **Correlación:** La amplitud máxima entre correlaciones de bloques cifrados debe ser mínima
  - **Probabilidad de propagación diferencial:** La probabilidad de poder deducir bloques desde los anteriores debe ser mínima
- **Complejidad algebraica:** El objeto matemático con el que se representa el dato debe de ser difuso y confuso...

Características preferentes

Esteganografía:

- **Naturaleza:** La naturaleza y atributos del objeto deben ser modificados en lo mas mínimo al momento de meterle el mensaje
- **Transformación:** El mensaje debe de ser transformado a manera que se asemeje a la función que modela el objeto

¿Entonces?

- La criptografía pide mas caos, asemejandose a un resultado estocástico.. en terminos de detección
- La esteganografía pide mayor cómputo, transformadas que nos indiquen dónde insertar un dato

Probabilidad

¿Qué pasará si medimos la media aritmética de datos cifrados?

$$\mu(M) = \frac{\sum_{i=0}^{n} x_i}{n}, \quad x_i \in \{0, 1, 2, ..., 255\} \quad \forall x_i \in M, n = |M|$$

Pruebas

Binarios

$ ./a.out /bin/mount
85.15
$ ./a.out /bin/sh
91.36
$ ./a.out /bin/cp
93.70
$./a.out /bin/ls
97.94
Vemos que los valores estan entre 80 y 100

Pruebas

Texto cifrado

$ openssl enc -aes256 -in /etc/services -out services.aes
enter aes-256-cbc encryption password:**josejose**
$ ./a.out services.aes
127.93
$ openssl enc -aes256 -in /etc/services -out services.aes
enter aes-256-cbc encryption password:**josejosUE**
$ ./a.out services.aes
128.11
$ dd if=/dev/urandom of=randfile bs=1024 count=1024
1024+0 records in
1024+0 records out
$ ./a.out randfile
127.43

Observaciones

- si $\mu(M) \approx 128$ entonces M es un texto cifrado o pseudo aleatorio
- Los datos cifrados son detectables con μ y hay más medidas y σ-algebras
- AES cumple todos los requerimientos de cifrado por NIST (National Institute of Standards and Technology)
- Lo cual implica que entre MEJOR sea un algoritmo de cifrado, es más fácil detectarlo.

Entropía

Existe otra medida mucho mejor... la cual nos proporciona mayor información... que nos permite medir si un conjunto de datos contiene redundancia. Esto nos puede servir para poder saber si un archivo es 'comprimible'.. por ejemplo... $(\Omega, 2^{\Omega}, \rho)$

$$\Gamma(X) = \sum_{i=1}^{n} \rho(x_i) \log_2 \left(\frac{1}{\rho(x_i)} \right) = -\sum_{i=1}^{n} \rho(x_i) \log_2 (\rho(x_i))$$

x_i son símbolos de $X, n = |\Omega|$ y $\rho(x)$ es una medida y $X \in 2^{\Omega}$

Donde 2^{Ω} es la σ-álgebra y Ω representa el conjunto de símbolos

La imágen de nuestros conjuntos bajo esta función será $\Gamma(X) \in [0,$

Esta función puede ser definida con cualquier medida inducida en una σ-álgebra

Pregunta

¿Por qué los valores de $\Gamma \in [0, 8]$ en nuestros archivos ?

Ejemplos archivos comunes

$ ./entropy /etc/passwd
5.30
$ perl -e "print 'X' x 10000;" > constante.dat
$ ./entropy constante.dat
0.00
$ ./entropy random.dat
8.00
$ ./entropy cifrado.aesbeck
8.00
$ ./entropy cifrado.aesopenssl
7.94

Ejemplos dumps de internet en un router muy concurrido

$ ./entropy dumpinet/dump1.raw
3.40
$ ./entropy dumpinet/dump2.raw
3.66
$ ./entropy dumpinet/dump3.raw
5.41
$ ./entropy dumpinet/dump4-ipsec.raw
6.05
$ ./entropy dumpinet/dump5-ipsec.raw
7.52
$ ./entropy dumpinet/dump6-ipsec.raw
7.24

Veamos unos dibujitos

Vamos a observar cómo se ven estos distintos tipos de conjuntos de datos con un programa que calcula una gráfica del comportamiento del archivo

Pseudo Filosofía

La vida contiene patrones, los lenguajes, la manera de pensarlos y disenarlos tienen una semántica, gramática, sintaxis. Y éstas no presentan **jamás** patrones entrópicos, ya que el fin con el que son disenados no contiene caos. Por lo tanto, cuando se presenta en internet entropía, es por alguna razón extrana.

Después de la barbacoa ahora los postrecitos

Métodos que explico a continuación muy brevemente sobre esteganografía básica con sus códigos respectivos serán mostrados

- LSB
- DCT (Discrete Cosine Transform) en JPEG

LSB (Least Significant Bit)

Ok, super sencillo... en mapas de bits por componentes , por ejemplo BMP o TIFF o cualquiera soportado por OpenCV lo puedes controlar por RGB (Red,Green,Blue)

Si cada pixel mide 24 bits tenemos un vector en 3 dimensiones cuy

componentes son las entradas (r,g,b) $\in \mathbb{F}_{2^8} \times \mathbb{F}_{2^8} \times \mathbb{F}_{2^8}$

No es snobbismo , pero tampoco es parte de la presentación verlo así...

pero les puede dar una idea, $\mathbb{F}_{2^8}$ es un campo de Galois..

Y se puede asociar un $\mathbb{Z}-$modulo por ser abeliano y ver la imágen

'quasi' espacio vectorial (sobre un anillo)

Después del debraye... regresamos

Un pixel es un entero de 24 bits... 0xPPQQRR por ejemplo 0xFF0000 es el rojo 0x0000FF es azul , entonces en binario se vería algo asi como 011010111111010110000 1101
Ahora imagina que quiero meter el bit K en ese pixel.
entonces mi pixel quedaría
011010111111010110000110K
Si K es 1 no se modifica el pixel , si K es 0 , se modifica 1 bit.

El pixel solo puede cambiar en una magnitud de 1/16777216

ya que son 2^{24} combinaciones de colores

Entropías

$ ./entropy imagen.bmp
7.88
$ ./entropy imagen-steg.bmp
7.88
La entropía no varía demasiado...

JPEG

Y que pasa con JPEG ? , ¿Por qué nadie puede? (o muy pocos lo logran) La respuesta es simple.

JPEG

Todo el mundo cree ser hacker, pero para ser 1337-H4X0R al menos debes saber que significa o para que sirve y cómo funciona esto:

$$F(\xi) := \int_{-\infty}^{\infty} f(x) e^{-2\pi i x \xi} \mathrm{d}x \quad \forall \xi \in \mathbb{R}$$

Dónde x es el momento y ξ es la frecuencia

Características básicas

JPEG usa una un caso especial de Fourier que es la transformada de coseno La cual se puede obtener fácilmente de la de Fourier expandiendo sus terminos con la identidad de Euler sobre la exponencial, y con propiedades trigonométricas de paridad en cosenos.

Deduzcamos la TC continua a través de Fourier

Si usamos que $e^{ix} = \cos(x) + i\sin(x)$ y suponemos que f es par

$$F(\xi) := \int_{-\infty}^{\infty} f(x)\cos(x2\pi\xi)\mathrm{d}x - i\int_{-\infty}^{\infty} f(x)\sin(x2\pi\xi)\mathrm{d}x \quad \forall\xi \in \mathbb{R}$$

Ahora... porque la parte compleja se va a 0 ? (eso queremos para obtener la TC)

TC continua

$f(x)\cos(x2\pi\xi)$ es par (¿porque?)

$f(x)\sin(x2\pi\xi)$ es impar

Y como se integra el seno de $(-\infty, \infty)$

Esto es simétrico con respecto al origen, y seno es impar..

Entonces: La parte compleja se va a 0 (Se complementa la simetría) y queda la parte real como todos sabemos:

$$F(\xi) := \int_{-\infty}^{\infty} f(x)\cos(x2\pi\xi)\mathrm{d}x$$

y Voila! analogamente obtendriamos la del seno si f fuera impar

- JPEG se fija que la luminancia es lo esencial para el ojo humano , entonces es a lo que le da mas énfasis, ya que cambios en color son menos detectables para el ojo que cambios de brillo.
- JPEG usa la TCD (Transformada de coseno discreta) ya que sabe juntar muy bién la energía en espacios mas chicos... y eso se aprovecha como 'compactificación'
- JPEG Guarda los coeficientes de la transformada que como todos sabemos se puede representar por una matriz (Nosotros somos hackers estudiosos)
- JPEG pierde datos al momento de cuantizar al redondear los valores flotantes (ahí se pierde la calidad)

Imagen

Examinemos la siguiente imágen:

Componentes de imágen YCbCr separados

¿Entonces?

Entonces si los colores se pierden, no nos sirve para guardar datos... ¿Qué es lo que nos sirve para guardar nuestros bits?

Final

Pués los coeficientes de la TCD!!
Despues de cuantizar (y perder datos) JPEG comprime con Huffman de una manera muy rara , organizando la matriz en zigzag ya que las frecuencias se organizan así, de mayor a menor así que generalmente hay muchos ceros al final ayuda a que se comprima muy bién.

Final

libjpeg es MUY cochina y difícil de usar... pero ésta tiene una gran función que te da los coeficientes de la TCD,
Veamos unos ejemplos con una implementación que oculta en JPEG sin modificarlo tanto, ni su tamano.

Contacto

¡Gracias!

Eduardo Ruiz Duarte
beck@math.co.ro
http://math.co.ro
http://b3ck.blogspot.com