Personal PNGSafeBox Project

¿De que estamos hablando?

Muchos estamos preocupados por la seguridad de nuestras claves, tarjetas de coordenadas, datos de tarjetas de crédito, semillas de nuestros bitcoins y cualquier información confidencial que nos conviene guardar y consultar de cuando en cuando.

Necesitamos consultarlos a menudo pero apuntar estos datos y llevarlos con nosotros no es muy aconsejable.

Esta es la razón que para poner en marcha este pequeño proyecto de protección sencilla de datos.

La idea es la siguiente:

1) Creamos una imagen del documento/dato a cifrar.
2) Lo cifraremos enmascarándolo dentro de un formato gráfico estándar.
3) Lo copiaremos – cifrado- en una App que podemos portar en nuestro teléfono móvil.
4) Lo podremos consultar usando una clave decidida solo por nosotros.
5) Controlaremos tanto la aplicación de cifrado/descifrado como la App móvil.

Como es lógico, no es un sistema garantizado al cien por cien pero nos puede aportar más fiabilidad que cualquier aplicación descargada de Internet por las siguientes razones:

* Se basa en un método de ofuscación sencillo y personalizado solo por nosotros.
* Nosotros vamos a decidir la complejidad del mismo y a desarrollar (o compilar) el programa.
* Nosotros vamos a instalar los ficheros securos en nuestro móvil (Android) directamente.
* Nosotros vamos a desarrollar nuestra App de control para Android.
* No necesitaremos permisos extra para accesos "extraños" a nuestro terminal.
* En resumen... controlaremos todo el proceso.

Este proyecto está soportado en dos aplicaciones/programas que – unidas - componen un simple pero efectivo sistema de protección personal de nuestros datos.

Podemos utilizar tan solo uno de ellos, pero su uso conjunto nos permite el control absoluto a la privacidad de nuestros datos.

Las dos partes del proyecto se resumen así:

Personal PNGSafeBox Project (I) - Aplicación Windows

Una aplicación desarrollada por nosotros – de forma gratuita - en VB.NET nos permitirá ofuscar los ficheros gráficos que contienen la información a proteger.

Dicha aplicación permite cifrar y descifrar dichas imágenes.

Personal PNGSafeBox Project (II) - Aplicación Android

Una App. Móvil desarrollada por nosotros – de forma gratuita – nos permitirá alojar las imágenes con información confidencial en nuestro terminal. Esta aplicación puede consultar dichos archivos (datos) de forma segura, descifrarlos y mostrarnos la información que necesitamos en pantalla y, por supuesto, seguirán cifrados dentro de nuestro móvil.

Incluso en el caso de que perdamos nuestro móvil los ficheros serían dicíciles de extraer al estar alojados nativamente junto a la aplicación y ,en el caso de un móvil "rooteado", se obtendrían tan solo unos ficheros gráficos ofuscados que no podrían verse salvo que se conozca exactamente el método de ofuccación utilizado.

Parte I Personal *PNGSafeBox* para Windows

Descripción general

El proyecto se basa en la modificación de ficheros gráficos (PNG) y cuya estructura se "tunea" para ser prácticamente inexpugnables.

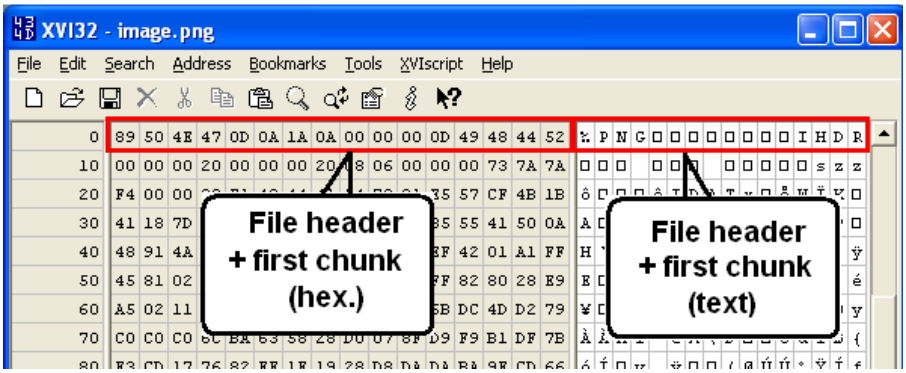
Por ejemplo, si queremos cifrar una imagen(foto) que contenga los datos de nuestra tarjeta de coordenadas tan solo tendremos que convertir la imagen a formato gráfico PNG y utilizar el programa comentado en este artículo.

El resultado será un fichero modificado de tal forma que no sea posible su visionado hasta que vuelva a ser "reordenado" de nuevo por el programa.

Algoritmo de cifrado utilizado.

Haremos uso de la estructura típica de un fichero gráfico PNG. Esta puede consultarse en la página oficial de dicho formato grafico (http://www.libpng.org/pub/png/spec/1.2/PNG-Contents.html), pero no es preciso su conocimiento detallado ya que nuestro programa se encarga de todo.

Un fichero gráfico PNG esta conformado una cabecera seguida de secciones que se denominan "chunks". La especificación de cada chunk no es objeto de este artículo, pero baste decir que utilizaremos parte de la cabecera del fichero PNG así como parte no esencial del primer chunk existente en el mismo.



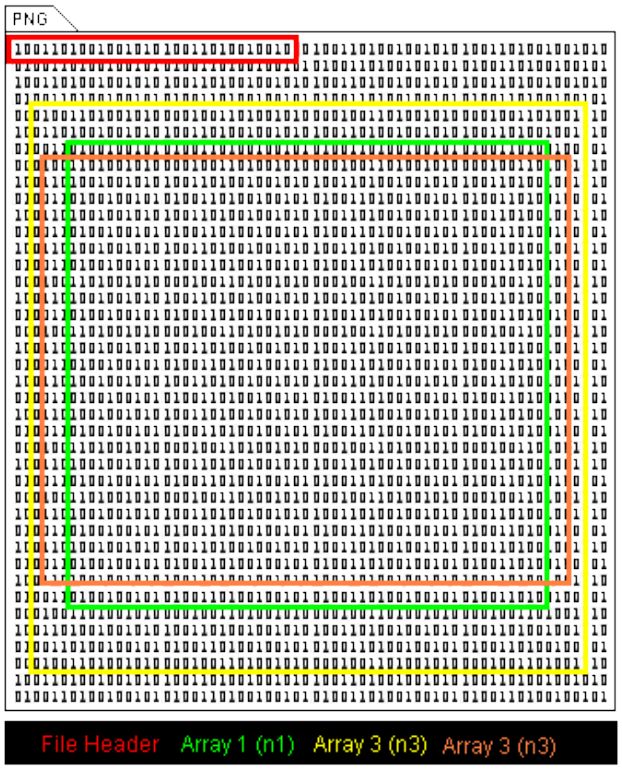
El algoritmo para cifrar la imagen – en formato PNG - a proteger realiza estos pasos:

1) Genera tres números alatorios de un rango especifico, N1, N2 y N3
2) Crea una matriz de bytes con datos del offset N1 a EOF-N1 del fichero gráfico.
3) Transpone dichos bytes y los reescribe al fichero.
4) Repite – dos veces más - similar procedimiento para los números N2 y N3.
5) Crea una cadena con la información /N1N2N3+

6) Cifra mediante triple DES la cadena utilizando nuestra KEY y IV
7) Escribe la cadena cifrada en una posición de cabecera del fichero PNG.
8) Realiza algunas modificaciones en la zona inicial de la cabecera del fichero PNG.

Una vez hechos estos pasos el fichero PNG obtenido queda altamante modificado y su visualización es imposible.

La única forma de visualizar de nuevo el fichero gráfico será utilizando el programa cifrador/descifrador para restaurar el PNG a su formato original.



Software utilizado para el Proyecto

Para desarrollar el programa hemos optado por una excelente herramienta de programación y que nos permite implementar el proyecto en VB.NET de forma totalmente gratuita.

Se trata del entorno Sharp Develop (http://www.icsharpcode.net/opensource/sd/Default.aspx)

En nuestro caso hemos optado por la versión 3.2.1 del citado software. Esta no es la última existente, pero nos ha resultado suficiente y, probablemente, será mucho más compatible con sistemas Windows que no estén actualizados al cien por cien.
El programa desarrollado es minimalista y tan solo persigue hacer lo que hace, es decir, cifrar y descifrar ficheros PNG.

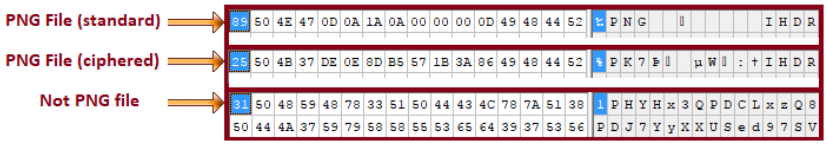
Podemos descargar el proyecto VB.NET desde el link que aparece al final del artículo o desde la sección de descargas de Webtronika.

Notas sobre el código del programa

A pesar de que el programa se ha desarrollado para que sea suficientemente auto-explicativo (contiene muchos comentarios) preferimos explicar aquí algunos de los aspectos mas destacados del mismo.

La primera rutina que ejecuta el programa es "**Initialize**", la cual – como su nombre indica – efectuará ciertas inicializaciones básicas. Una vez efectuadas el control recae en el botón "**Open File**" , el cual nos permite elegir el fichero a tratar (cifrar o descifrar).

La rutina asociada al botón llamará a una verificación – función **IsCiphered ( )** - sobre el fichero seleccionado. Mediante la verificación del primer byte de dicho fichero el programa conocerá si se trata de un fichero PNG estándar o bien un fichero previamente cifrado por nuestro programa. En el caso de que el primer byte no coincida con estas dos posibilidades se calificará dicho fichero de inválido para ser tratado por el programa.



La rutina de codificación - **CodeFile ( )** - ejecutará el algoritmo de codificación comentado en lineas anteriores. Cabe destacar que la cadena a codificar se conforma por las tres posiciones a guardar (cada una de ellas forzada a dos posiciones) y un par de símbolos ("/" y "+") al principio y final de la misma que ayudarán en posteriores verificaciones, y por supuesto usando la codificación Base64. Esto hace que la cadena final sea de 8 posiciones.

Por ejemplo, si las tres posiciones obtenidas aleatoriamente fueran 3, 23 y 41 la cadena **s0** a cifrar quedará finalmente como **/032341+**

Podemos ver que la rutina de descodificación ejecuta un proceso similar pero al transponer las matrices del fichero lo hace – lógicamente – en sentido inverso.

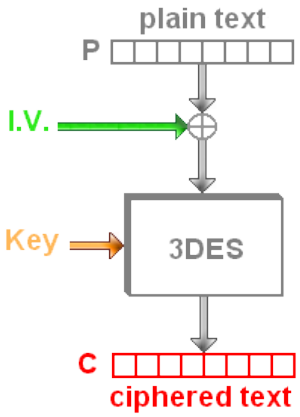
Además de esto, la rutina de descodificación ejecuta la subrurina **ChkDecodeString ( )** que controla que el contenido de la cadena descifrada cumple con el formato esperado, con lo comentado para **s0**. Para ello hace uso de varias comprobaciones :

* Longitud de la cadena s0 es de 8 posiciones.
* Los caracteres de la cadena s0 pertenecen a la base numérica Base64.
* Los 6 caracteres centrales son de tipo numérico.
* Los caracteres inicial y final son / y +.

La rutina utilizada para generar los tres números aleatorios se ha prefijado a unos márgenes que eviten que el número obtenido (offset a utilizar posteriormente) sea mayor de 15, con lo evitamos "tocar" la cabecera del fichero y que será donde alojaremos posteriormente la información cifrada (mediante 3DES) de las tres posiciones aleatorias.

Es de destacar que la codificación del valor (cadena) que contiene la información sobre las tres posiciones aleatorias generadas se realiza mediante un cifrado triple DES operado en modo CBC /

no Padding. Para ello necesitaremos utilizar una KEY de cifrado (16 bytes) y un vector de inicialización (8 bytes). Dadas las longitudes de ambas cadenas se ha preferido dar la posibilidad al usuario de modificar desde el frontal del programa tan solo el valor prefijado del vector de inicialización (I.V.) ya que dada su menor longitud puede ser más fácil de recordar por el usuario a modo de "password".



El valor de la KEY de cifrado se ha prefijado en el programa (en la cabecera), pero puede ser modificado a nuestro gusto. Tan solo hemos de respetar que la cadena utilizada use tan solo caracteres pertenecientes a la codificación Base64.

Value	Char	Value	Char	Value	Char	Value	Char
0	A	16	Q	32	g	48	w
1	B	17	R	33	h	49	x
2	C	18	S	34	i	50	y
3	D	19	T	35	j	51	z
4	E	20	U	36	k	52	0
5	F	21	V	37	l	53	1
6	G	22	W	38	m	54	2
7	H	23	X	39	n	55	3
8	I	24	Y	40	o	56	4
9	J	25	Z	41	p	57	5
10	K	26	a	42	q	58	6
11	L	27	b	43	r	59	7
12	M	28	c	44	s	60	8
13	N	29	d	45	t	61	9
14	O	30	e	46	u	62	+
15	P	31	f	47	v	63	/

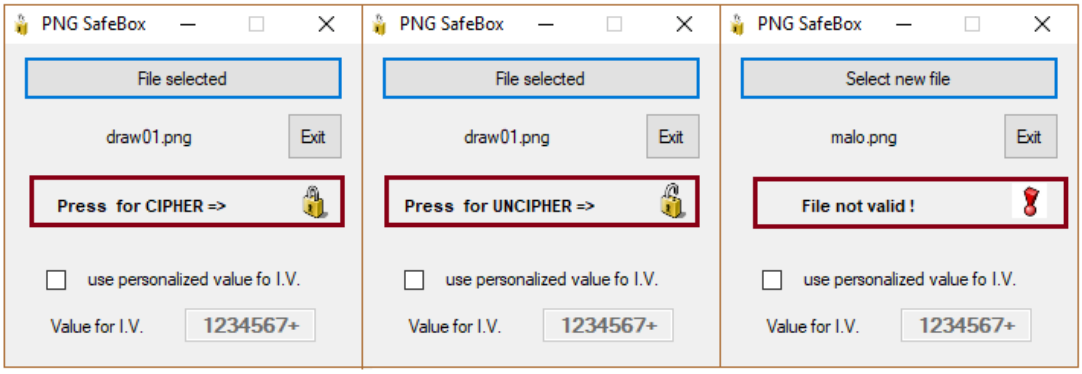
La rutina de trasposición utilizada - **Transpose (F, P)** - es la misma tanto en la codificación comoi en la descodificación del fichero. La única diferencia estriba en que es llamada tres veces y dichas llamadas se realizan en orden inversoi en la rutina de descodificación.

La rutina **ChkKey ( )** se encarga de verificar en tiempo real que el vector de inicialización seleccionado manualmente (I.V.) cumpla con la longitud y formato adecuado. Si es así la casilla IV se colorea en verde, y en rojo si no lo es.

Manejo del programa

La utilzación del programa no tiene secreto alguno. Se ha prefijado para que se ejecute en una ventana muy reducida y centrada en la pantalla.

Para cifrar un archivo tan solo hemos de seleccionar el mismo. Inmediatamente el nombre del fichero aparecerá en el frontal del programa y pulsando sobre el icono del candado se cifrará o descifrará el fichero elegido.



Al abrir el fichero se ejecuta un análisis previo del mismo y, como hemos comentado, dependiendo del valor del primer byte del fichero el candado nos dará la opción de cifrar, descifrar o nos indicará que el fichero selecionado no es válido.

Los valores de **byte 0** que la rutina puede encontrarse son :

- 0x89 => Fichero no cifrado (PNG estándar).
- 0x25 => Fichero PNG cifrado por nuestro programa.
- Otros => Fichero de formato no reconocido.