

OPENPUFF v4.00 STEGANOGRAFIA & WATERMARKING

Nascondere dati e marking, semplice, sicuro e gratuito

Ing. Cosimo Oliboni, Italia, 2012

Inviare i vostri suggerimenti, commenti, segnalazioni, richieste
a oliboni@embeddedsw.net

OPENPUFF HOMEPAGE



[NOTE LEGALI](#)



[CARATTERISTICHE: PERCHÈ QUESTO PROGRAMMA STEGANOGRAFICO È DIFFERENTE DAGLI ALTRI?](#)



[CARATTERISTICHE: ARCHITETTURA DEL PROGRAMMA](#)



[CARATTERISTICHE: CODIFICA ADATTIVA E RESISTENZA ALLA STEGANALISI](#)



[CARATTERISTICHE: MULTI CRITTOGRAFIA E OFFUSCAMENTO DATI](#)



[COSA È LA STEGANOGRAFIA?](#)



[COSA È LA STEGANOGRAFIA NEGABILE?](#)



[COSA È IL MARKING?](#)



[FORMATI SUPPORTATI IN DETTAGLIO](#)



[SUGGERIMENTI PER RISULTATI MIGLIORI](#)



[OPZIONI: LIVELLO DI SELEZIONE BIT](#)



[DATA HIDING IN DETTAGLIO](#)



[DATA UNHIDING IN DETTAGLIO](#)



[MARK SETTING IN DETTAGLIO](#)



[MARK CHECKING IN DETTAGLIO](#)



[DATA & MARK ERASING IN DETTAGLIO](#)



NOTE LEGALI

Ricordate: questo programma non è stato scritto per uso illegale. L'uso di questo programma in violazione delle leggi del vostro paese è assolutamente proibito. L'autore declina qualsiasi responsabilità conseguente dall'uso improprio di questo programma.

Né codice né formati coperti da brevetto sono stati inseriti in questo programma.

Questo programma, diversamente dai codec (librerie di codifica/decodifica), non manipola dati video o audio. Esclusivamente i bit ausiliari (bit inutilizzati dello stream) vengono trattati. Ogni altro elemento viene semplicemente copiato inalterato.

QUESTO È UN SOFTWARE FREWARE

Questo software è rilasciato con licenza [CC BY-ND 3.0](#)

Siete liberi di copiare, distribuire, modificare e fare uso commerciale di questo software alle seguenti condizioni:

- Dovete citare l'autore (e detentore del copyright): [Eng. Cosimo Oliboni](#)
- Dovete fornire un link alla Homepage dell'autore: [EMBEDDEDSW.NET](#)

[INDIETRO](#)



Caratteristiche: perchè questo programma steganografico è differente dagli altri?

OpenPuff è un programma professionale di steganografia, con caratteristiche uniche che non troverete in nessun'altro programma gratuito o commerciale. OpenPuff è 100% gratuito e adatto alla trasmissione nascosta di dati altamente sensibili.

[COSA È LA STEGANOGRAFIA?](#)

Una panoramica delle sue caratteristiche

- [CATENE DI CARRIER]
I dati vengono suddivisi fra i vari carrier. Solo la sequenza corretta di carrier ne permette il recupero. Potrete inoltre nascondere fino a 256Mb di dati, avendo abbastanza carrier a disposizione. L'ultimo carrier verrà riempito con bit random per renderlo indistinguibile dagli altri.
- [FORMATI SUPPORTATI]
Immagini, file audio, video, flash, adobe.
[FORMATI SUPPORTATI IN DETTAGLIO](#)
- [LIVELLI DI SICUREZZA]
I dati, prima di essere iniettati nei carrier, sono crittografati (1), sottoposti a scrambling (2), a whitening (3) e codificati (4).
[CARATTERISTICHE: ARCHITETTURA DEL PROGRAMMA](#)
 - [LIVELLO 1 - MULTI CRITTOGRAFIA MODERNA]
Un insieme di 16 algoritmi di crittografia a 256bit, moderni e open-source è stato unito per formare un algoritmo di multi crittografia a doppia password (256bit+256bit).
 - [LIVELLO 2 - SCRAMBLING BASATO SU CSPRNG]
I dati crittografati sono sempre sottoposti a scrambling per spezzare qualsiasi struttura residua dello stream. Viene inizializzato un nuovo generatore di numeri pseudo-casuali crittograficamente sicuro (CSPRNG) con una terza password (256bit) e i dati vengono mischiati globalmente con indici random.
 - [LIVELLO 3 - WHITENING BASATO SU CSPRNG]
I dati sottoposti a scrambling sono sempre mischiati ad una grande quantità di rumore, proveniente da un CSPRNG indipendente inizializzato con entropia hardware.
[OPZIONI: LIVELLO DI SELEZIONE BIT](#)
 - [LIVELLO 4 - CODIFICA ADATTIVA NON-LINEARE]
I dati sottoposti a whitening sono sempre codificati usando una funzione non-lineare che usa come input anche i bit originali dei carrier. I carrier modificati subiranno meno cambiamenti e supereranno molti test steganalitici (p.e.: χ^2 test).
[CARATTERISTICHE: CODIFICA ADATTIVA E RESISTENZA STEGANALITICA](#)
 - [SICUREZZA EXTRA - STEGANOGRAFIA NEGABILE]
I dati altamente sensibili possono essere protetti usando dei dati meno sensibili come esca.
[COSA È LA STEGANOGRAFIA NEGABILE?](#)

- [CODICE SORGENTE]

Questo programma si basa sulla libreria [LIBOBFUSCATE](#), indipendente dal sistema e open-source. Gli utenti e gli sviluppatori sono assolutamente liberi di utilizzare la libreria di base (100% del codice di crittografia e offuscamento), leggerla e modificarla.

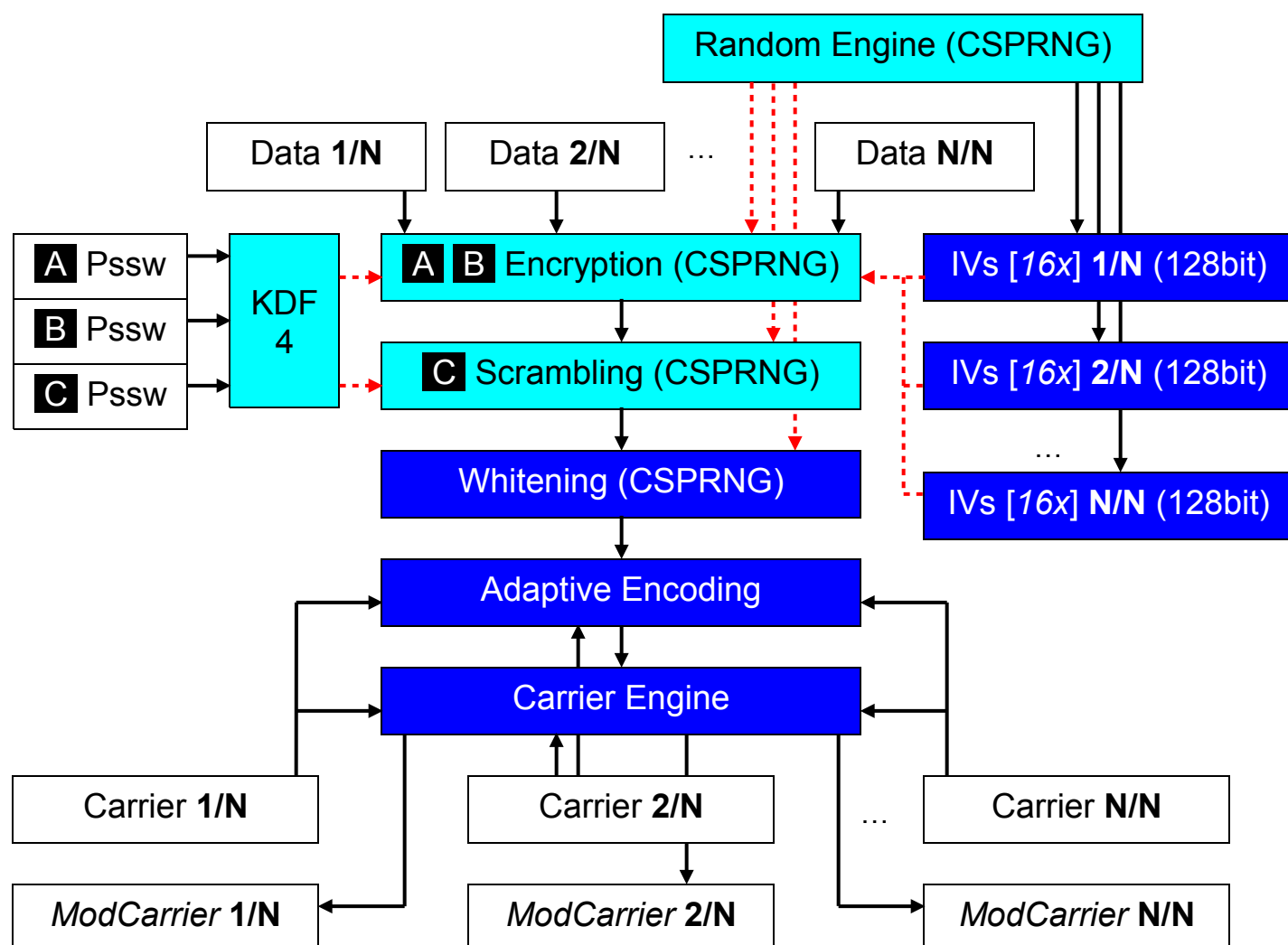
Siete gentilmente pregati di inviarmi i porting/upgrade/personalizzazioni/sw derivati di libObfuscate, per analizzarli e aggiungerli alla homepage del progetto. Un repository ufficiale, centrale e aggiornato eviterà dispersione e irraggiungibilità del codice derivato dal progetto.

[INDIETRO](#)

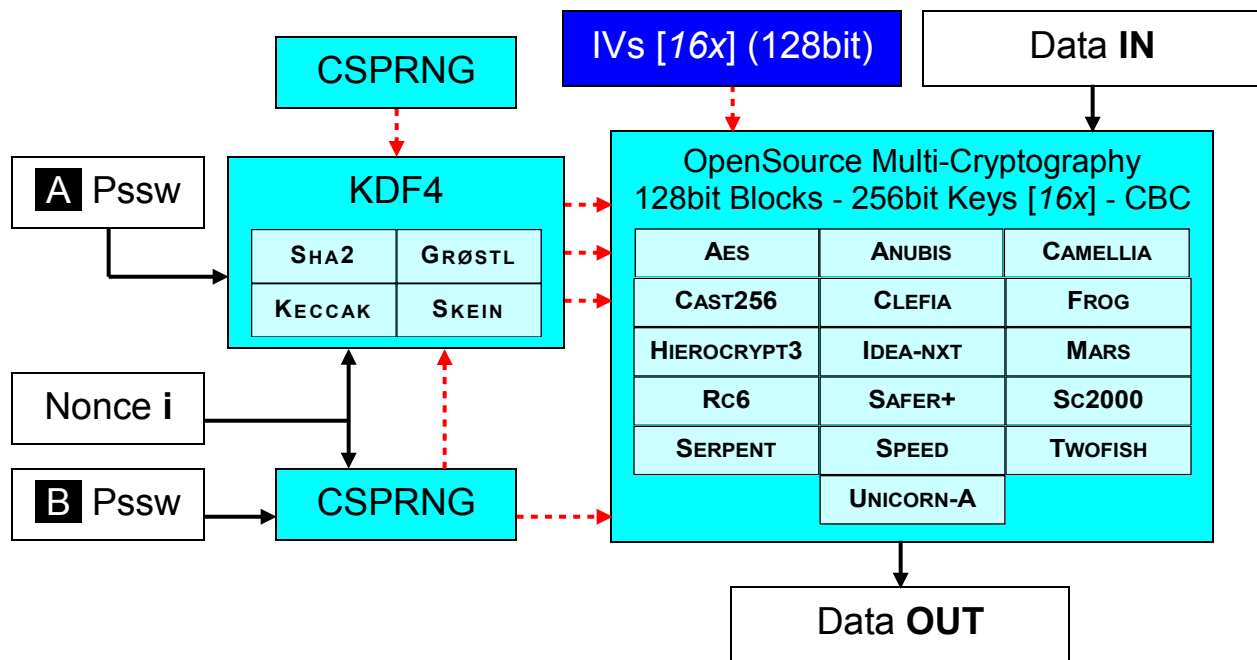


Una descrizione globale dell'architettura di alto livello di OpenPuff

- i dati sono divisi fra i carrier
- ogni carrier è associato ad un array di vettori random di inizializzazione a 128bit (**IVs**)
- le password testuali (32 caratteri = 256bit) sono associate (**KDF4**) a delle password esadecimali
- i dati vengono prima crittografati con **CHIAVI** a 256bit (**A**) (**B**), usando la multi crittografia
- i dati crittografati vengono poi sottoposti a scrambling, con una terza chiave a 256bit (**C**), per spezzare qualsiasi struttura residua dello stream
- i dati sottoposti a scrambling vengono poi sottoposti a whitening (= mischiati con rumore random)
- i dati sottoposti a whitening vengono codificati usando una funzione che usa come input anche i bit originali dei carrier
- i carrier modificati ricevono lo stream processato



OpenPuff implementa la multi crittografia (un tipo avanzato di [CRITTOGRAFIA PROBABILISTICA](#)) unendo 16 moderni algoritmi crittografici a blocchi open-source, scelti fra [AES-PROCESS](#), [NESSIE-PROCESS](#) e [CRYPTREC-PROCESS](#). Il Cypher-Block-Chaining (CBC) ha il ruolo di wrapper per questi algoritmi a blocchi, permettendo loro di comportarsi come algoritmi a stream.



Il setup della multi crittografia è un processo in 4 passi

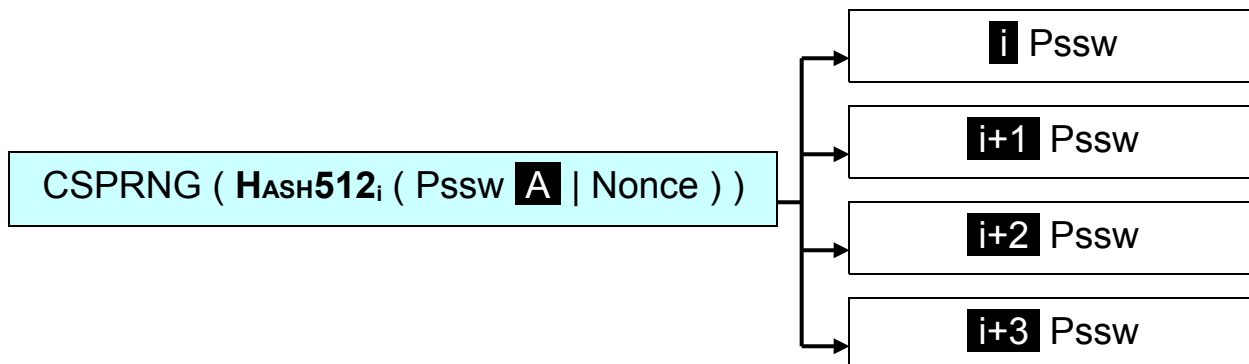
- un array di vettori random di inizializzazione (16 x 128bit) è associato ad ogni carrier
- un motore pseudo random (CSprNG) viene inizializzato usando la password (B) e un [NONCE](#) (l'indice del carrier)
- la password (A) e il nonce sono estesi (KDF4) usando 4 moderni algoritmi open-source di hashing a 512bit, provenienti dallo [SHA2](#) e dallo [SHA3](#). Ogni hash genera quattro chiavi a 256bit

$$Psw (1) | (2) | (3) | (4) = Rand (Sha2 (Psw (A) | Nonce))$$

$$Psw (5) | (6) | (7) | (8) = Rand (Grøstl (Psw (A) | Nonce))$$

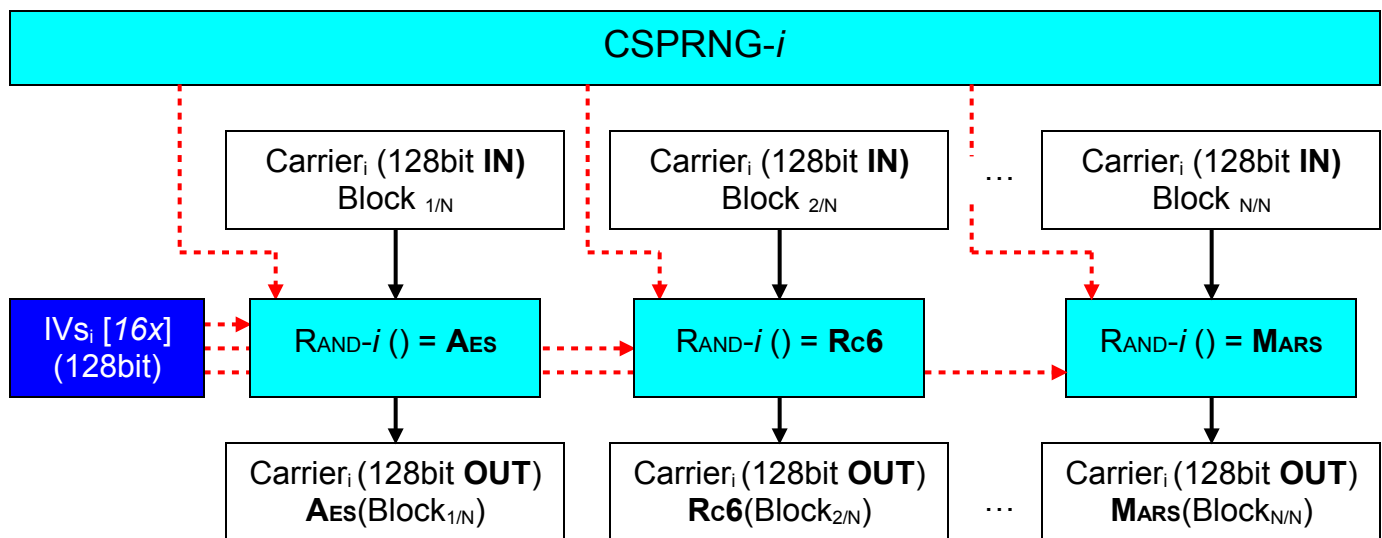
$$Psw (9) | (10) | (11) | (12) = Rand (Keccak (Psw (A) | Nonce))$$

$$Psw (13) | (14) | (15) | (16) = Rand (Skein (Psw (A) | Nonce))$$
- l'array di chiavi risultante (16 x 256bit) è associato ad ogni algoritmo usando il CSprNG



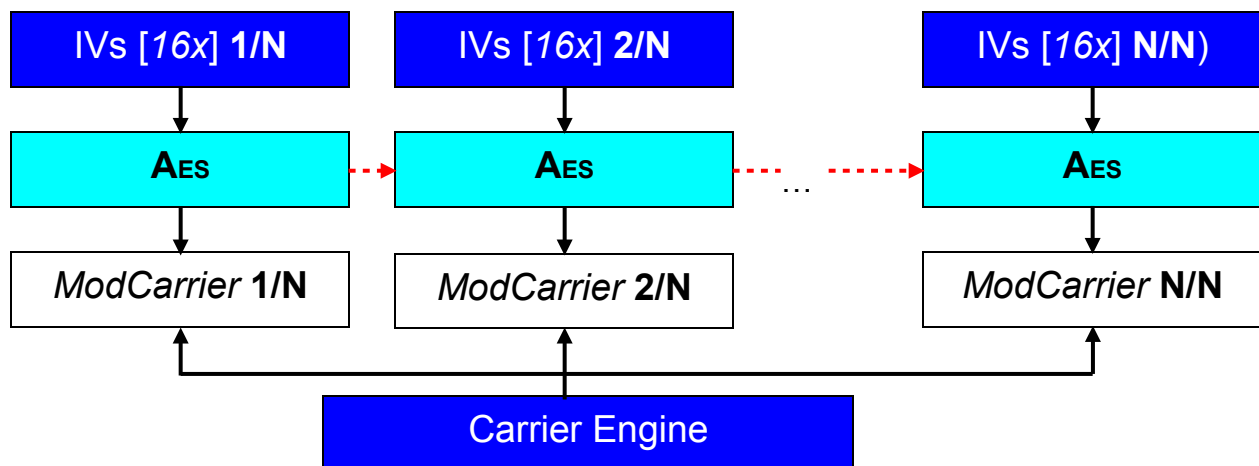
La crittografia è un processo in più passi

- ogni carrier è sottoposto ad un setup indipendente
 $CarrierSetup_i = \{ IVs_i, CSPRNG_i, Keys_i \}$
- ogni algoritmo è sottoposto ad un setup indipendente
 $Cipher_j = \{ IV_j, Key_j \}$
- ogni blocco di dati è processato con un algoritmo diverso, scelto usando il CSPRNG
 $Carrier_i CryptedBlock_k = r \leftarrow Rand-i () ; Cipher_r (IV_r, Key_r, Carrier_i Block_k)$

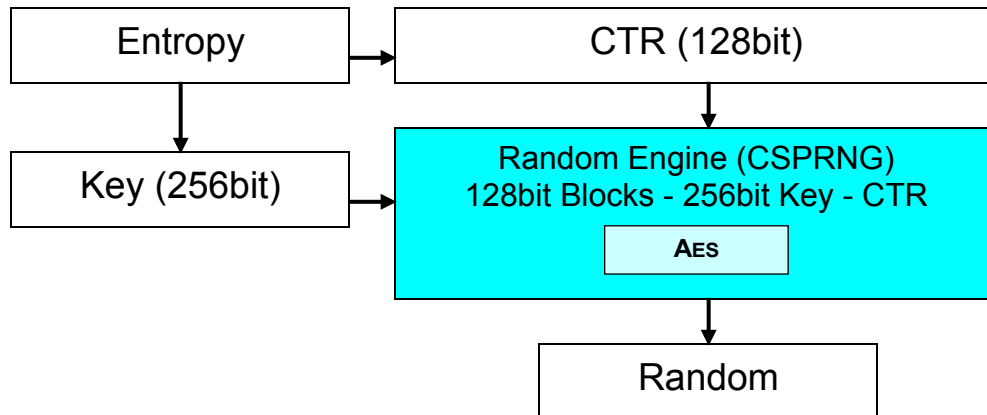


I carrier modificati ricevono

- una copia crittografata (AES) del loro array di vettori di inizializzazione
 $CryptedIVs_n = Crypt (IVs_n, CryptedIVs_{n-1})$
- i dati processati

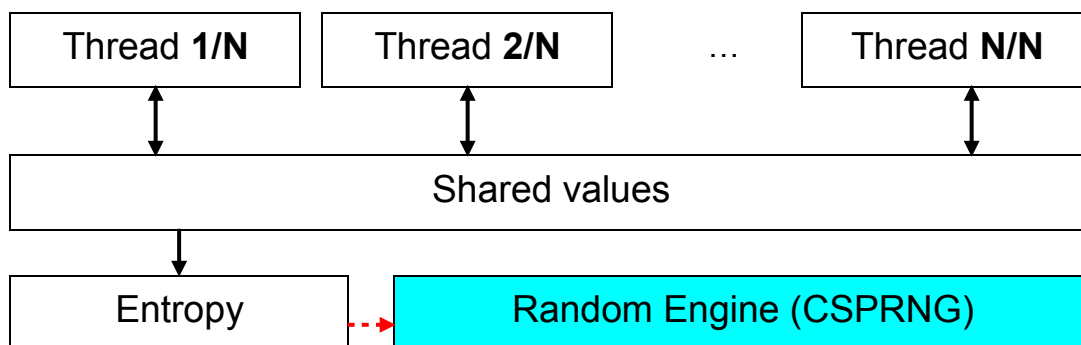


OpenPuff implementa un generatore di numeri pseudo-casuali crittograficamente sicuro ([CSPRNG](#)) usando la crittografia AES-256. Gli algoritmi a blocchi sicuri implementati in Counter-Mode (CTR) si comportano, per costruzione, come un motore random.



È stata aggiunta anche una buona sorgente hardware di entropia iniziale, indipendente da librerie o API di sistema. I thread sono sempre eseguiti dal SO in una sequenza imprevedibile (dovuta ad un inevitabile assenza di precisione nella temporizzazione), consentendo di ottenere facilmente una quantità significativa di [EXECUTION RACE CONDITION](#). N thread eseguono in parallelo, incrementando e decrementando delle variabili condivise che, dopo un po', si trasformano in valori random.

Le sequenze di ripetizione a lungo termine (inevitabili) sono gestite usando il CTR reseeding (ogni T chiamate al motore random).



Sono stati condotti test sulla resistenza alla statistica del CSPRNG e del multi wrapper, usando il noto [PSEUDORANDOM NUMBER SEQUENCE TEST PROGRAM - ENT](#).

I risultati forniti derivano da campioni di 64Kb, 128Kb, ... 256Mb:

- resistenza al test dell'entropia dei bit

>7.9999xx / 8.000000	<i>referimento: >7.9</i>
----------------------	-----------------------------

- resistenza al test della compressione (riduzione della dimensione dopo la compressione):

0%	<i>referimento: <1%</i>
----	----------------------------

- resistenza al test della distribuzione chi-quadro:

20% < deviazione < 80%	<i>referimento: >10%, <90%</i>
------------------------	--------------------------------------

- resistenza al test del valore medio:

127.4x / 127.5	<i>referimento: >127, <128</i>
----------------	--------------------------------------

- resistenza al test Monte Carlo:

errore < 0.01%	<i>referimento: < 1%</i>
----------------	-----------------------------

- resistenza al test della correlazione seriale:

< 0.0001	<i>referimento: < 0.01</i>
----------	-------------------------------

[INDIETRO](#)



Caratteristiche: codifica adattiva e resistenza alla steganalisi

Sicurezza, performance e resistenza alla steganalisi sono obiettivi contrastanti.

[Sicurezza vs. Performance]: Whitening

- Pro: assicura una maggior sicurezza dei dati
- Pro: consente la steganografia negabile
- **Con1:** *richiede molti carrier bit extra*

[Sicurezza vs. Steganalisi]: Crittografia + Whitening

- Pro: assicura una maggior sicurezza dei dati
- **Con2:** *il risultato statistico simile ai dati random marca i carrier come "sospetti"*

Dobbiamo quindi preoccuparci della [RESISTENZA ALLA STEGANALISI](#) di OpenPuff? I dati, prima di essere iniettati nei carrier, sono crittografati (1), sottoposti a scrambling (2) e whitening (3). Questi 3 passaggi trasformano una piccola quantità di dati nascosti in un grande blocco di dati sospetti?

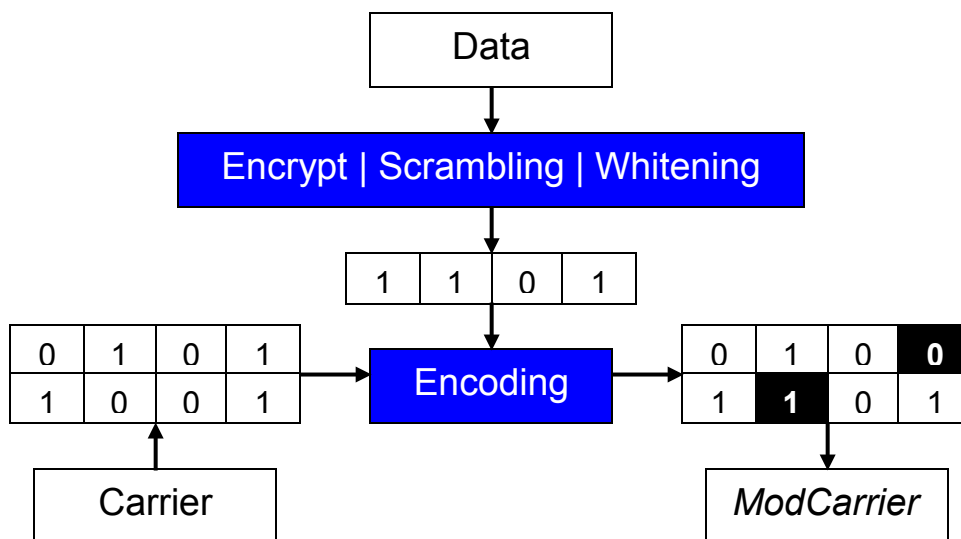
In fondo al processo è stato aggiunto un nuovo livello di sicurezza. I dati sottoposti a whitening sono sempre codificati usando una funzione non-lineare che usa come input anche i bit originali dei carrier. I carrier modificati subiranno meno cambiamenti (**Con1**) e supereranno molti test steganalitici (**Con2**).

["DEFENDING AGAINST STATISTICAL STEGANALYSIS"](#) (Niels Provos)

["CONSTRUCTING GOOD COVERING CODES FOR APPLICATIONS IN STEGANOGRAPHY"](#) (Jessica Fridrich)

L'implementazione fornita della codifica è una funzione non pubblicata (costruita da zero) che assicura

- dipendenza dell'output dalla password
- elevata (50%) efficienza di inserimento
- basso (<20%) tasso di modifica



[INDIETRO](#)



FAQ 1: Perché non è stata implementata una crittografia standard AES-256 or RSA-1024?

La moderna crittografia open-source

- è stata studiata approfonditamente e analizzata dalla comunità scientifica
- è largamente accettata come lo strumento più sicuro per proteggere i dati
- soddisfa praticamente ogni necessità *standard* di sicurezza

OpenPuff non appoggia nessuna [TEORIA DELLA COSPIRAZIONE](#) contro la nostra privacy ([BACKDOOR SEGRETE](#), design crittografici intenzionalmente deboli, ...). Non c'è nessuna ragione per non avere fiducia nella moderna crittografia pubblicamente disponibile (sebbene qualche vecchio cifrario sia già stato [VIOLATO](#)).

Gli utilizzatori della steganografia, comunque, molto probabilmente nascondono dati molto sensibili, con una necessità *insolitamente alta* di sicurezza. I loro segreti hanno bisogno di subire un approfondito processo di [OFFUSCAMENTO](#) dei dati per poter sopravvivere *più a lungo* alle indagini forensi e agli attacchi brute-force potenziati da hardware specializzato.

FAQ 2: La multi crittografia è simile alla cifratura multipla?

La multi crittografia è qualcosa di molto diverso dalla [CIFRATURA MULTIPLA](#) (crittografare più di una volta). Non ci sono opinioni largamente condivise riguardo all'affidabilità della cifratura multipla. Si pensa che sia:

- [MIGLIORE](#) della cifratura singola
- [DEBOLE](#) come il cifrario più debole della coda/processo di crittografia
- **peggiore** della cifratura singola

OpenPuff appoggia l'ultima tesi (peggiore) e non crittografa mai dati già crittografati.

FAQ 3: La multi crittografia è simile alla crittografia random/polimorfica?

La crittografia random, alias. [CRITTOGRAFIA POLIMORFICA](#), è una ben nota [CRITTOGRAFIA FRAUDOLENTA](#). La multi crittografia è qualcosa di molto diverso e non aspira mai a costruire cifrari migliori, random o generati dinamicamente.

OpenPuff si basa unicamente sulla moderna crittografia stabile e open-source.

FAQ 4: La multi crittografia è migliore della crittografia standard?

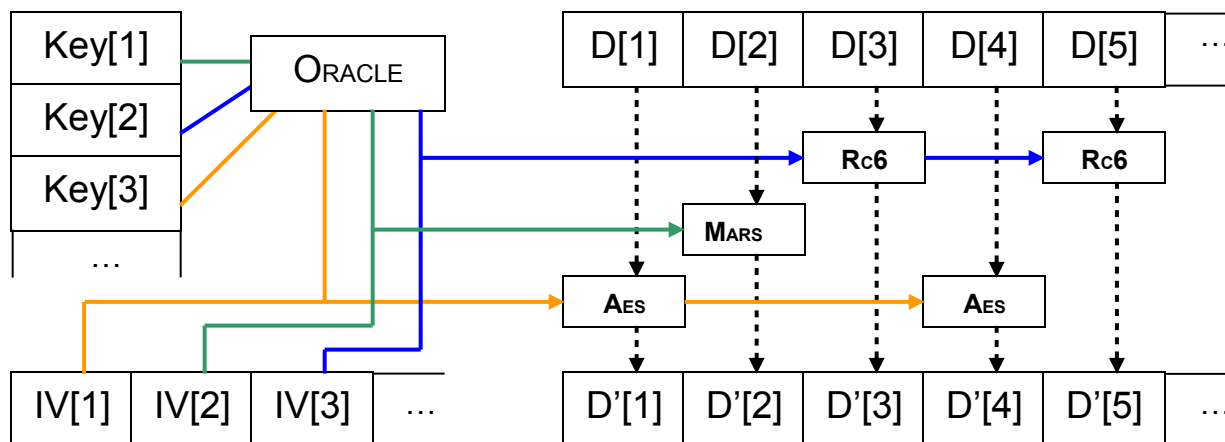
Una *casa* è migliore di un *mattone*? No. La casa è una *superstruttura* e il mattone è un *materiale*.

La *multi crittografia* è migliore della *crittografia*? No. La multi crittografia è parte di un *processo* di offuscamento dati e la crittografia è un *componente*.

Crittografia	=	Mattone
Multi crittografia	=	Piano
Processo di offuscamento	=	Casa

La multi crittografia è un livello di offuscamento

- la crittografia e il CSPRNG ricevono due password indipendenti
- la crittografia e il CSPRNG ricevono anche un nonce dipendente dall'indice del carrie
- ogni cifrario implementato riceve un IV e una chiave differenti
- il CSPRNG si comporta come un [ORACOLO](#) che alimenta il motore crittografico durante tutte le sue scelte (quale chiave associare a quale cifrario, quale cifrario applicare a quale blocco di dati, ...)

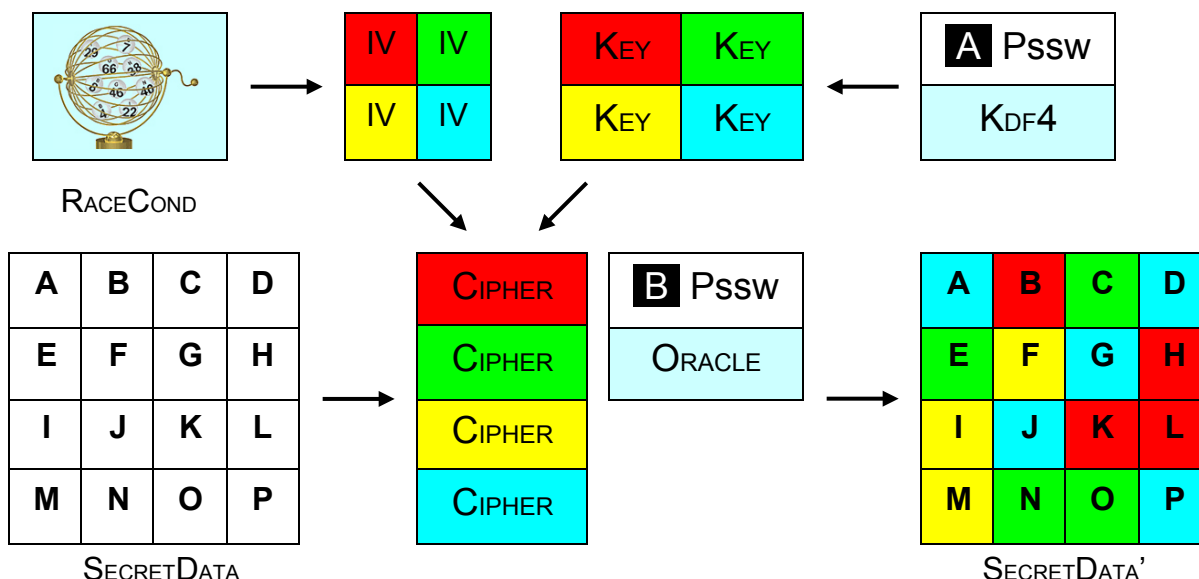


FAQ 5: L'offuscamento dati è migliore della crittografia standard?

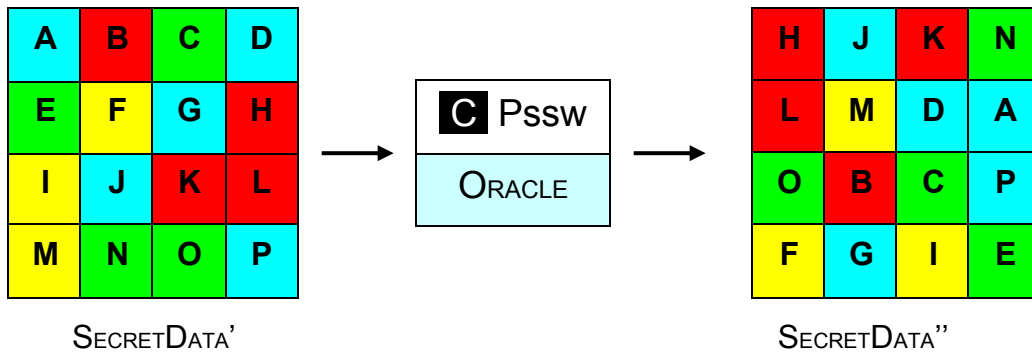
L'offuscamento dati non ha nessuna pretesa di [INVIOLEABILITÀ](#) (da considerare sempre come un sintomo di inganno o [CRITTOGRAFIA FRAUDOLENTA](#)). È comunque possibile gestire il problema del bisogno *insolitamente alto* di sicurezza in modo effettivo e costruttivo (secondo il [PRINCIPIO DI KERCKHOFF'S](#)), come un problema (*rallentare gli attaccanti* il più possibile) da ingegnerizzare

- connettere trasformazioni di offuscamento diverse
- evitare di applicare ripetutamente la stessa trasformazione
- affidarsi unicamente a risorse open-source
- applicare qualche trasformazione globale, invertibile solo via software

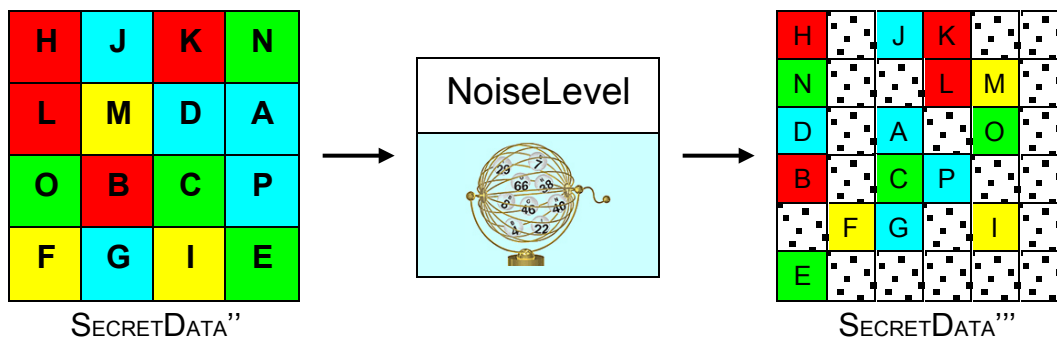
[Round 1 – MULTI CRITTOGRAFIA (TRASFORMAZIONE LOCALE)]



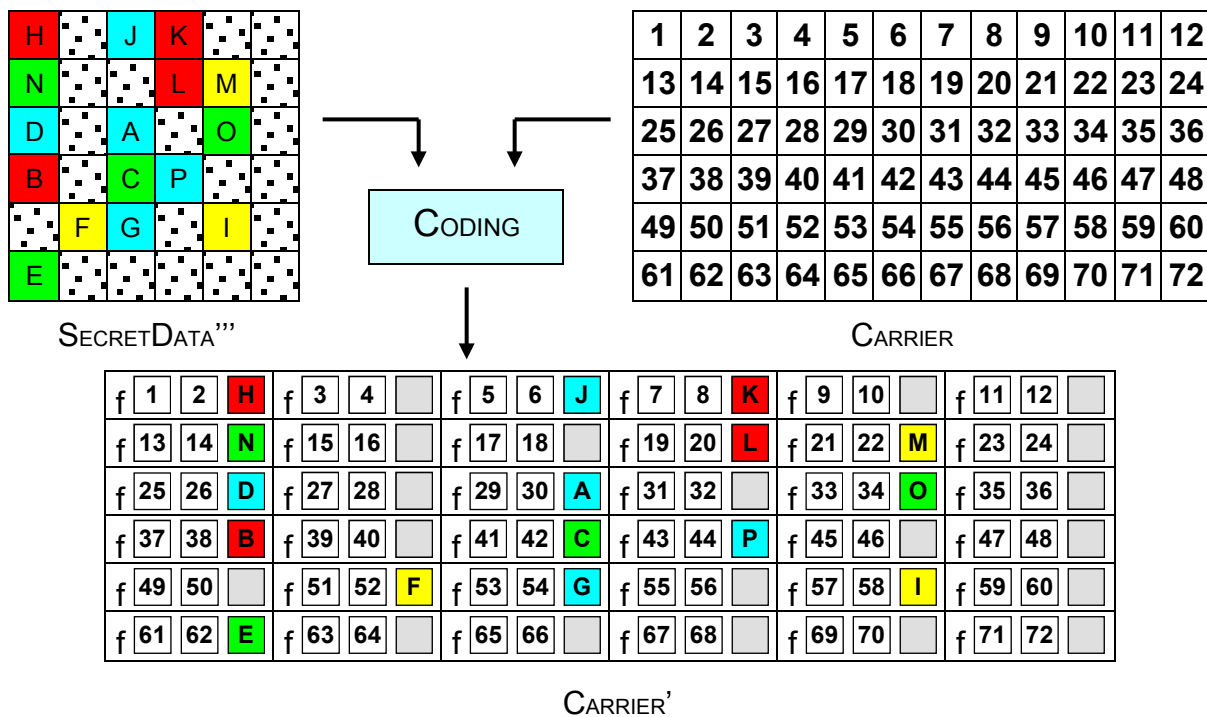
[Round 2 – SCRAMBLING (TRASFORMAZIONE GLOBALE)]



[Round 3 – WHITENING (TRASFORMAZIONE GLOBALE)]



[Round 4 – ENCODING (TRASFORMAZIONE LOCALE)]



[CARATTERISTICHE: ARCHITETTURA DEL PROGRAMMA](#)

[INDIETRO](#)



COSA È LA STEGANOGRAFIA?

È un [METODO ORIGINALE](#) per nascondere dati all'interno di altri file, chiamati **carrier**. I carrier modificati avranno lo stesso aspetto degli originali, senza alcun cambiamento percettibile. I migliori carrier sono video, immagini e file audio, dato che chiunque può inviarli/riceverli/scaricarli senza destare sospetti.

Il processo di steganografia di OpenPuff è altamente selettivo e adattivo, con un payload minimo per ogni carrier. I carrier con un contenuto nascosto massimo inferiore del payload minimo saranno scartati.

- +256B → IV
- +16B → un blocco di crittografia

[CARATTERISTICHE: ARCHITETTURA DEL PROGRAMMA](#)

Non esiste alcun limite di CARRIER byte per il processo di marking.

[COSA È IL MARKING?](#)

PERCHÉ DOVREI AVERNE BISOGNO?

Non avete bisogno di questo metodo se i vostri dati

- possono venire esplicitamente inviati o salvati in formato alterato o sospetto.

Potreste essere interessati a questo metodo se i vostri dati

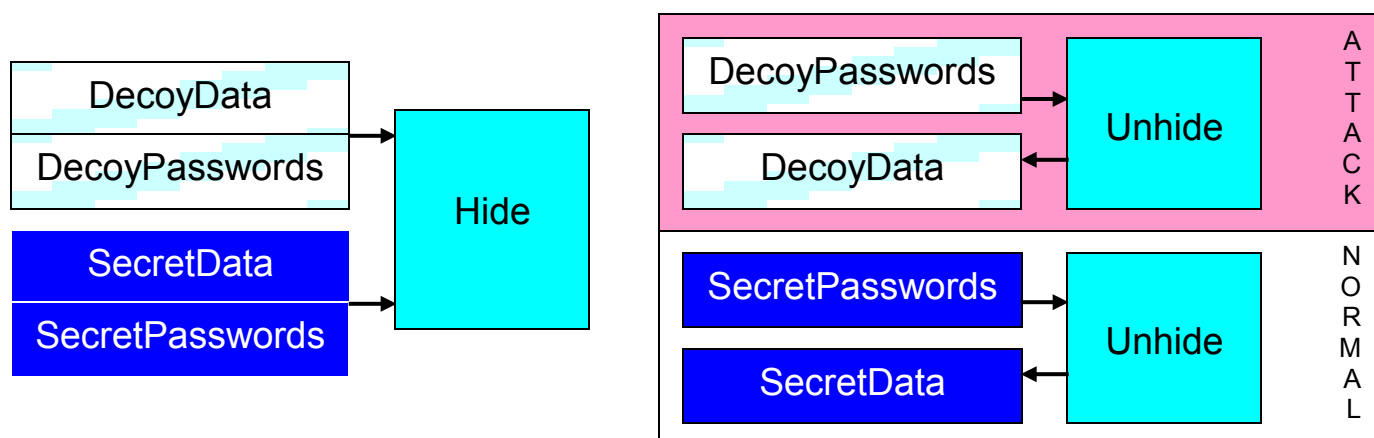
- hanno bisogno di venire nascosti senza destare sospetti.
- devono essere facilmente accessibili da chiunque, ma recuperabili solo da coloro a conoscenza del vostro metodo segreto.

[INDIETRO](#)

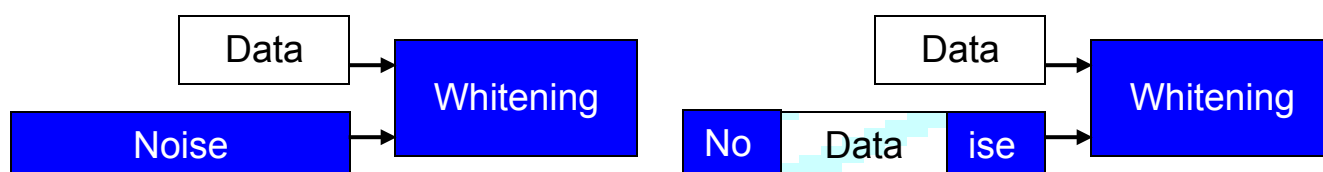


COSA È LA STEGANOGRAFIA NEGABILE?

La [CRITTORGRAFIA/STEGANOGRAFIA NEGABILE](#), è una tecnica basata sull'uso di un'esca che permette di negare in maniera convincente di stare nascondendo dati sensibili, anche se gli attaccanti possono dimostrare che si sta nascondendo qualcosa. Basta semplicemente fornire un'esca sacrificabile che **plausibilmente** deve rimanere confidenziale. Verrà rivelata all'attaccante, sostenendo che questa è l'unico contenuto.



Come è possibile? I dati crittografati e sottoposti a scrambling, prima di essere iniettati nei carrier, sono sottoposti a whitening ([CARATTERISTICHE: ARCHITETTURA DEL PROGRAMMA](#)) con una grande quantità di rumore ([OPZIONI: LIVELLO DI SELEZIONE BIT](#)). I dati esca possono sostituire un po' del rumore senza compromettere le proprietà finali di [RESISTENZA ALLA CRITTANALISI](#).

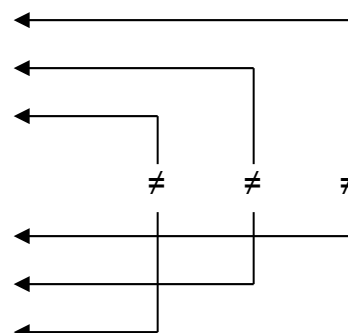


I dati sensibili e i dati esca sono crittografati usando password differenti. Si devono scegliere due diversi insiemi di diverse password.

Esempio:

Sensibile data: Password (A) "FirstDataPssw1"
 Password (B) "SecondDataPssw2"
 Password (C) "AnotherDataPssw3"
 (A ∩ B) 70%, (A ∩ C) 67%, (B ∩ C) 68%, [HAMMING DISTANCE](#) ≥ 25%

Decoy data: Password (A') "FirstDecoyPssw1"
 Password (B') "SecondDecoyPssw2"
 Password (C') "AnotherDecoyPssw3"
 (A' ∩ B') 72%, (A' ∩ C') 60%, (B' ∩ C') 70%, [HAMMING DISTANCE](#) ≥ 25%



Le password devono essere diverse (a livello di bit) e lunghe almeno 8 caratteri.

Esempio: "DataPsw1" (A) "DataPsw2" (B) "DataPsw3" (C)

(A) 01000100 01100001 01110100 01100001 01010000 01110011 01110011 01110111 00110001
 (B) 01000100 01100001 01110100 01100001 01010000 01110011 01110011 01110111 00110010
 (C) 01000100 01100001 01110100 01100001 01010000 01110011 01110011 01110111 00110011
 (A ∩ B) 98%, (A ∩ C) 99%, (B ∩ C) 99%, [HAMMING DISTANCE](#) < 25% **KO**

Esempio: "FirstDataPsw1" (A) "SecondDataPsw2" (B) "AnotherDataPsw3" (C)

(A) 01000110 01101001 01110010 01110011 01110100 01000100 01100001 01110100 01100001 ...
 (B) 01010011 01100101 01100011 01101111 01101110 01100100 01000100 01100001 01110100 ...
 (C) 01000001 01101110 01101111 01110100 01101000 01100101 01110010 01000100 01100001 ...
 (A ∩ B) 70%, (A ∩ C) 67%, (B ∩ C) 68%, [HAMMING DISTANCE](#) ≥ 25% **OK**

Verranno richiesti

- due **diversi** insiemi di diverse password
- un file di dati sensibili
- un file di dati esca **compatibile** (per dimensione) con i dati sensibili

$$\sum_{k \in \{1, N-1\}} used_carrier_bytes(carr_k) < Sizeof(Decoy) \leq \sum_{k \in \{1, N\}} used_carrier_bytes(carr_k)$$

Esempio:

Carriers	Carrier bytes	SensibleData	DecoyData
+Carr (1/N)	32	X	Used
...	2688	X	Used
+Carr (N-1/N)	48	X	Used
+Carr (N/N)	64		Not used
	Total = 2832	Total = 2795	2720 < Size ≤ 2768

[INDIETRO](#)



COSA È IL MARKING?

Per Marking si intende l'azione di firmare un file con il vostro marchio di copyright (meglio noto come [WATERMARKING](#)). Questo programma lo fa in modo steganografico, all'interno di video, immagini e file audio. Il vostro marchio di copyright risulterà invisibile, ma accessibile da parte di chiunque (usando questo programma), dal momento che non sarà protetto da password.

PERCHÉ DOVREI AVERNE BISOGNO?

Non avete bisogno di questo metodo se il vostro marchio di copyright

- deve risultare chiaramente visibile
- deve essere indipendente dai dati, quindi capace di sopravvivere alle operazioni di editing

Potrete essere interessati a questo metodo se il vostro marchio di copyright


- deve essere invisibile
- deve dipendere dai dati grafici/audio, quindi impossibilitato a sopravvivere alle operazioni di editing
- deve essere accessibile da chiunque (usando questo programma)

Un uso di questo metodo potrebbe essere: inserimento di un marchio di copyright invisibile all'interno di file registrati che devono essere pubblicamente condivisi. Le copie manipolate illegalmente potranno forse somigliare agli originali, ma perderanno totalmente/parzialmente il marchio invisibile.

[INDIETRO](#)



FORMATI SUPPORTATI IN DETTAGLIO

 Immagini:	BMP , JPG , PCX , PNG , TGA
 Audio:	AIFF , MP3 , NEXT/SUN , WAV
 Video:	3GP , FLV , MP4 , MPG , SWF , VOB
 Flash-Adobe:	PDF

I carrier manterranno il loro formato

- [in: 32 bit per piano TGA, out: 32 bit per piano TGA]
- [in: Stereo WAV, out: Stereo WAV]
- [in: RGB+Alpha BMP, out: RGB+Alpha BMP]

ecc...

I tag/chunk aggiuntivi e i byte extra saranno integralmente copiati senza modifiche.

Non eseguite nessuna nuova operazione sui carrier modificati. I loro carrier bit verrebbero molto probabilmente alterati.

[INDIETRO](#)

IMMAGINI BMP (MICROSOFT)

- Estensioni note: **.BMP, *.DIB*
- 24/32 bit per pixel
- Mono/RGB/RGB+Alpha
- Fino alla versione 5

[INDIETRO](#)

IMMAGINI JPG (JOINT PHOTOGRAPHIC EXPERTS GROUP)

- Estensioni note: **.JPG, *.JPE, *.JPEG, *.JFIF*
- 8 bit per piano
- 1-4 piani per pixel, ex: Mono/RGB/YCbCr/YCbCrK/CMY/CMYK
- Baseline lossy DCT-jfif con compressione Huffman
- Piani con allineamento indipendente h2v2 (4:4), h1v2 (4:2), h2v1 (4:2), h1v1 (4:1)

[INDIETRO](#)

IMMAGINI PCX (ZSOFT)

- Estensioni note: **.PCX*
- 24 bit per pixel Mono/RGB
- Compresso/Non compresso

[INDIETRO](#)

IMMAGINI PNG (PORTABLE NETWORK GRAPHICS)

- Estensioni note: **.PNG*
- 8/16 bit per piano
- Mono/RGB/Mono+Alpha/RGB+Alpha
- Interlacciato/Lineare

[INDIETRO](#)

IMMAGINI TGA (TARGA TRUEVISION)

- Estensioni note: **.TGA, *.VDA, *.ICB, *.VST*
- Mono-8 bit per pixel o RGB/RGB+Alpha-24/32 bit per pixel
- Compresso/Non compresso

[INDIETRO](#)

AUDIO AIFF (AUDIO INTERCHANGE FILE FORMAT)

- Estensioni note: **.AIF, *.AIFF*
- 16 bit per campione
- Mono/Stereo/Multi canale
- Lineare, non compresso

[INDIETRO](#)

AUDIO MP3 (FRAUNHOFER INSTITUT)

- Estensioni note: **.MP3*
- MPG 1/MPG 2/MPG 2.5 Layer III
- Bitrate fisso/variabile
- Mono/Dual Channel/Joint Stereo/Stereo
- ID Tagged

[INDIETRO](#)

AUDIO NEXT/SUN (SUN & NEXT)

- Estensioni note: **.AU, *.SND*
- 16 bit per campione
- Mono/Stereo/Multi canale
- Lineare, non compresso

[INDIETRO](#)

AUDIO WAV (MICROSOFT)

- Estensioni note: **.WAV, *.WAVE*
- 16 bit per campione
- Mono/Stereo/Multi canale
- PCM, non compresso

[INDIETRO](#)

VIDEO 3GP (3RD GENERATION PARTNERSHIP PROGRAM)

- Estensioni note: **.3GP, *.3GPP, *.3G2, *.3GP2*
- Fino alla versione 10
- Supporto indipendente dal codec
- Fino a 32 tracce

[INDIETRO](#)

VIDEO ADOBE FLV (FLASH VIDEO)

- Estensioni note: [*.FLV](#), [*.F4V](#), [*.F4P](#), [*.F4A](#), [*.F4B](#)
- Fino alla versione 10
- Supporto indipendente dal codec
- Analisi delle tracce audio [MP3](#)

[INDIETRO](#)

VIDEO MP4 (MOTION PICTURE EXPERTS GROUP)

- Estensioni note: [*.MP4](#), [*.MPG4](#), [*.MPEG4](#), [*.M4A](#), [*.M4V](#), [*.MP4A](#), [*.MP4V](#)
- Fino alla specifica ISO/IEC 14496-12:2008
- Supporto indipendente dal codec
- Fino a 32 tracce

[INDIETRO](#)

VIDEO MPG (MOTION PICTURE EXPERTS GROUP)

- Estensioni note: [*.MPG](#), [*.MPEG](#), [*.MPA](#), [*.MPV](#), [*.MP1](#), [*.MPG1](#), [*.M1A](#), [*.M1V](#), [*.MP1A](#), [*.MP1V](#), [*.MP2](#), [*.MPG2](#), [*.M2A](#), [*.M2V](#), [*.MP2A](#), [*.MP2V](#)
- Mpeg I Systems - fino alla specifica ISO/IEC 11172-1:1999
- Mpeg II Systems - fino alla specifica ISO/IEC 13818-1:2007
- Supporto indipendente dal codec

[INDIETRO](#)

VIDEO ADOBE SWF (SHOCKWAVE FLASH)

- Estensioni note: [*.SWF](#)
- Fino alla versione 10
- Supporto indipendente dal codec
- Analisi delle tracce audio [MP3](#)

[INDIETRO](#)

VIDEO VOB (DVD - VIDEO OBJECT)

- Estensioni note: [*.VOB](#)
- Mpeg II Systems - fino alla specifica ISO/IEC 13818-1:2007
- Supporto indipendente dal codec

[INDIETRO](#)

FILE ADOBE PDF (PORTABLE DOCUMENT FORMAT)

- Estensioni note: **.PDF*
- Fino alla specifica ISO/IEC 32000-1:2008
- Supporto indipendente dalla revisione

[INDIETRO](#)



SUGGERIMENTI PER RISULTATI MIGLIORI

CATENE DI CARRIER:

Nascondete i vostri dati all'interno di catene singole o multiple di carrier, aggiungendo carrier in ordine imprevedibile. I tentativi di recupero da parte di estranei non autorizzati cresceranno in complessità.

Esempio di carrier singolo: (Semplice, Recupero veloce, Non sicuro)

+MieiDati >> John.mp3

Esempio di catena singola: (Complessità media, Recupero medio, Sicuro)

+MieiDati >> Bear.jpg | Arrow.png | John.bmp | ...

Esempio di catene multiple: (Complessità massima, Recupero lento, Più sicuro)

+MieiDati (1/n) >> Bear.jpg | Arrow.png | John.bmp | ...

...

+MieiDati (2/n) >> Zoo.tga | Arrow.png | Beep.wav | ...

PASSWORD:

Usate password lunghe (>16 caratteri) facili da ricordare, cambiandole di volta in volta.

LIVELLO DI SELEZIONE BIT DEI CARRIER:

Usate sempre livelli diversi per ogni operazione. I tentativi di recupero da parte di estranei non autorizzati cresceranno in complessità.

Esempio:

Operazione 1:

- **Aiff: Basso**
- **BMP: Molto basso**
- **JPG: Massimo**

...

Operazione 2:

- **AIFF: Medio**
- **BMP: Basso**
- **JPG: Minimo**

...

UN SISTEMA COMPLETO POTREBBE ESSERE...

- Nascondere i vostri dati all'interno di numerose catene complesse (centinaia di carrier, in ordine random non alfabetico), ognuna con una diversa password di 32 caratteri
- Salvare tutte le impostazioni all'interno di un unico carrier "indice"

Esempio:

+MyData (1/n)	[carrier1 ... carrier100] [VeryLongPasswords1] [BitsSelectionLevel1]
...	
+MyData (n/n)	[carrier1 ... carrier100] [VeryLongPasswordsN] [BitsSelectionLevelN]

Un carrier "indice" assolutamente non sospetto (password fissa + livello di selezione bit fisso) contenente un file di testo che riassume

- i nomi dei carrier e l'ordine
- le password
- i livelli di selezione bit

[INDIETRO](#)



OPZIONI: LIVELLO DI SELEZIONE BIT

(<i>Minimum</i>)	1/8 dati, 7/8 whitening.
(<i>Very Low</i>)	1/7 dati, 6/7 whitening.
(<i>Low</i>)	1/6 dati, 5/6 whitening.
(<i>Medium</i>)	1/5 dati, 4/5 whitening.
(<i>High</i>)	1/4 dati, 3/4 whitening.
(<i>Very High</i>)	1/3 dati, 2/3 whitening.
(<i>Maximum</i>)	1/2 dati, 1/2 whitening.

[INDIETRO](#)



DATA HIDING IN DETTAGLIO

INIZIO:



([Hide](#)) [Vai al pannello Data Hiding](#)

Selezionare *Hide*.

PASSO 1:

(1) Insert 3 uncorrelated data passwords (Min: 8, Max: 32)

Cryptography (A) (B)

Scrambling (C) Enable (B) ☒ (C) ☒

Passwords check: H(A, B) H(A, C) H(B, C) = { 2%, 1%, 1% }

H(X, Y) = Hamming distance (X)(Y) >= 25%

(1) Insert 3 uncorrelated data passwords (Min: 8, Max: 32)

Cryptography (A) (B)

Scrambling (C) Enable (B) ☒ (C) ☒

Passwords check: H(A, B) H(A, C) H(B, C) = { 30%, 33%, 32% }

H(X, Y) = Hamming distance (X)(Y) >= 25%

(Cryptography A)	La prima password (chiavi crittografiche)
(Cryptography B)	La seconda password (CSPRNG crittografico)
(Scrambling C)	La terza password (CSPRNG scrambling)
(Enable B)	Abilita/disabilita la seconda password
(Enable C)	Abilita/disabilita la terza password

Inserire tre distinte password. Le password devono essere diverse (a livello di bit) e lunghe almeno 8 caratteri. Il tipo e il numero delle password può essere personalizzato facilmente disabilitando la seconda (B) e/o la terza (C) password. Le password disabilitate saranno impostate come la prima (A) password.

Esempio: "DataPsw1" (A) "DataPsw2" (B) "DataPsw3" (C)

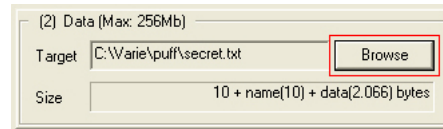
(A) 01000100 01100001 01110100 01100001 01010000 01110011 01110011 01110111 00110001
 (B) 01000100 01100001 01110100 01100001 01010000 01110011 01110011 01110111 00110010
 (C) 01000100 01100001 01110100 01100001 01010000 01110011 01110011 01110111 00110011
 (A ∩ B) 98%, (A ∩ C) 99%, (B ∩ C) 99%, [HAMMING DISTANCE](#) < 25% **KO**

Esempio: "FirstDataPsw1" (A) "SecondDataPsw2" (B) "AnotherDataPsw3" (C)

(A) 01000110 01101001 01110010 01110011 01110100 01000100 01100001 01110100 01100001 ...
 (B) 01010011 01100101 01100011 01101111 01101110 01100100 01000100 01100001 01110100 ...
 (C) 01000001 01101110 01101111 01110100 01101000 01100101 01110010 01000100 01100001 ...
 (A ∩ B) 70%, (A ∩ C) 67%, (B ∩ C) 68%, [HAMMING DISTANCE](#) ≥ 25% **OK**

[SUGGERIMENTI PER RISULTATI MIGLIORI](#)
[COSA È LA STEGANOGRAFIA NEGABILE?](#)

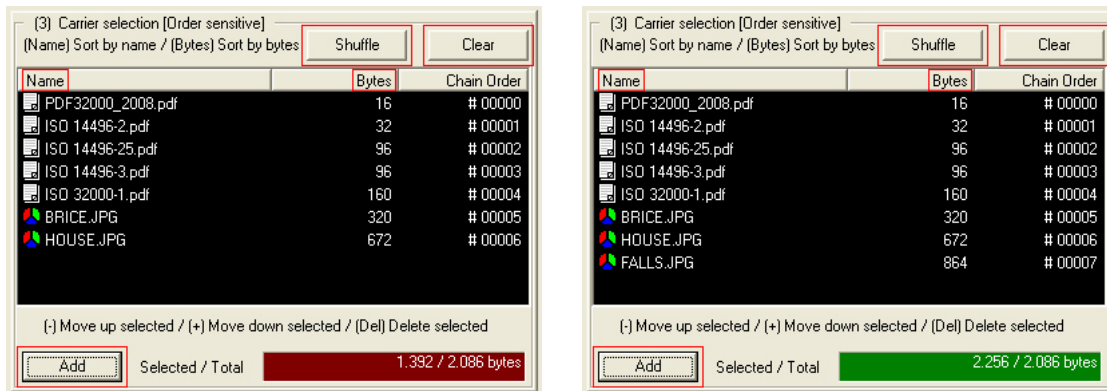
PASSO 2:



(Browse)	Selezionare un file
----------------------------	---------------------

Selezionare i dati segreti che si vogliono nascondere (tipicamente un archivio zip/rar/...).

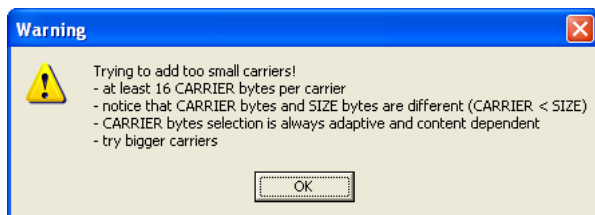
PASSO 3:



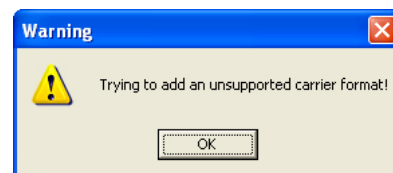
(Shuffle)	Shuffle random di tutti i carrier
(Clear)	Scartare tutti i carrier
(Add)	Aggiungere nuovi carrier alla lista
(Name)/(Bits)	Ordinare i carrier per nome/bt
(+)/(-)	Muovere in su/giù i carrier selezionati
(Del)	Eliminare i carrier selezionati

Finché *selected bytes* < *total bytes* provare ad

- aggiungere nuovi carrier
- aumentare il livello di selezione bit



(I)



(II)

Alcuni carrier non saranno inclusi a causa di vincoli del processo di steganografia

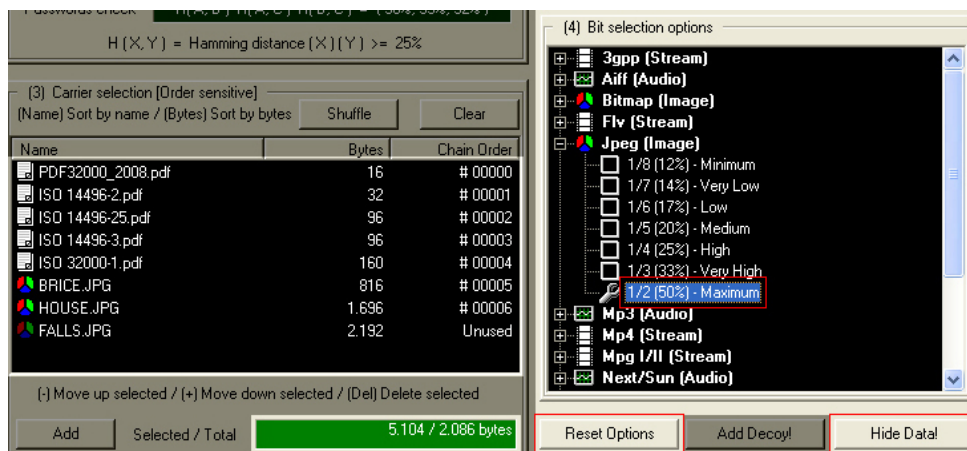
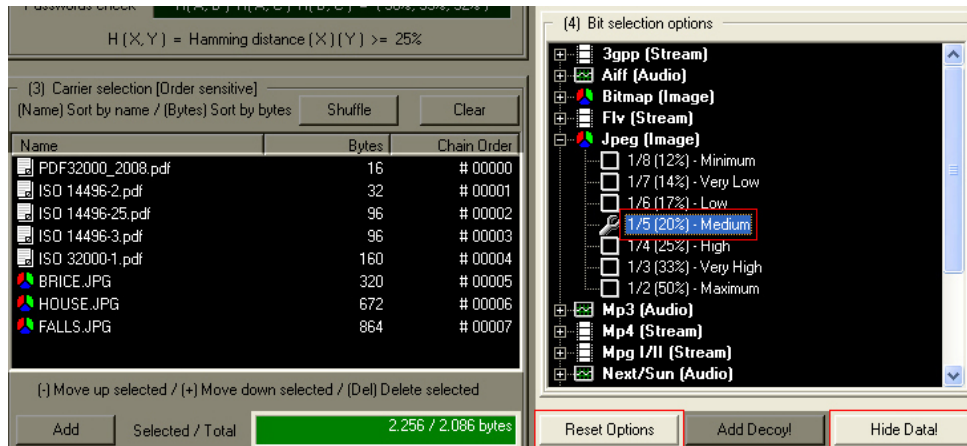
- (I) carrier byte insufficienti (carrier byte < dimensione del carrier)

[COSA È LA STEGANOGRAFIA?](#)

- (II) formato non supportato

[FORMATI SUPPORTATI IN DETTAGLIO](#)

PASSO 4:



(Reset Options)	Reset dei livelli di selezione bit
(Add Decoy!)	Aggiungere un'esca (steganografia negabile)
(Hide!)	Inizio dell'operazione di hide

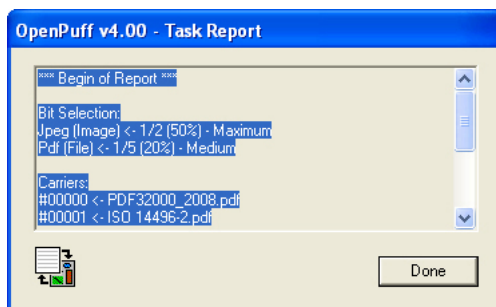
Dopo

- aver inserito due volte la stessa password, minimo 8 caratteri
- aver selezionato un file non vuoto da nascondere
- aver aggiunto abbastanza bit di carrier
- aver aggiunto un'esca (opzionale)

iniziare l'operazione di hide

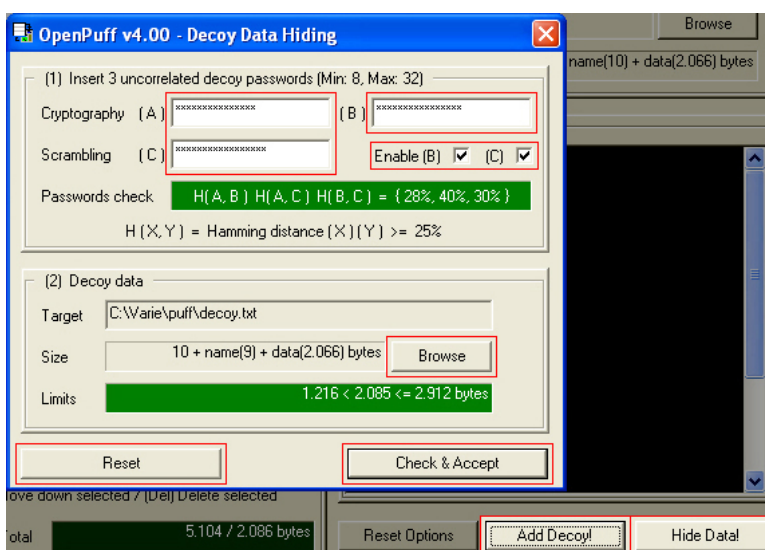
[OPZIONI: LIVELLO DI SELEZIONE BIT](#)

RAPPORTO FINALE:



Il rapporto finale riassume tutte le informazioni necessarie a recuperare i dati.

PASSO 4 – (FACOLTATIVO):



(Cryptography A)	La prima password (chiavi crittografiche)
(Cryptography B)	La seconda password (CSPRNG crittografico)
(Scrambling C)	La terza password (CSPRNG scrambling)
(Enable B)	Abilita/disabilita la seconda password
(Enable C)	Abilita/disabilita la terza password
(Browse)	Selezionare un file
(Reset)	Reset della password e del file
(Check & Accept)	Verifica la correlazione della password e la dimensione del file

Si possono inoltre aggiungere delle password e dei dati esca

- le password esca devono essere **diverse** fra loro e diverse da quelle sensibili
- il tipo e il numero delle password esca può essere personalizzato come per le password sensibili
- i dati esca devono essere **compatibili** (per dimensione) con i dati sensibili

$$\sum_{k \in \{1, N-1\}} \text{used_carrier_bytes}(carr_k) < \text{Sizeof}(\text{Decoy}) \leq \sum_{k \in \{1, N\}} \text{used_carrier_bytes}(carr_k)$$

[COSA È LA STEGANOGRAFIA NEGABILE?](#)

[INDIETRO](#)



DATA UNHIDING IN DETTAGLIO

INIZIO:



(Unhide)	Vai al pannello Data Unhiding
--------------------------	-------------------------------

Selezionare *Unhide*.

PASSO 1:

(Cryptography A)	La prima password (chiavi crittografiche)
(Cryptography B)	La seconda password (CSPRNG crittografico)
(Scrambling C)	La terza password (CSPRNG scrambling)
(Enable B)	Abilita/disabilita la seconda password
(Enable C)	Abilita/disabilita la terza password

Inserire le vostre password (secrete per estrarre i dati segreti, esca per estrarre i dati esca), abilitando solo quelle usate al momento dell'operazione di hide.

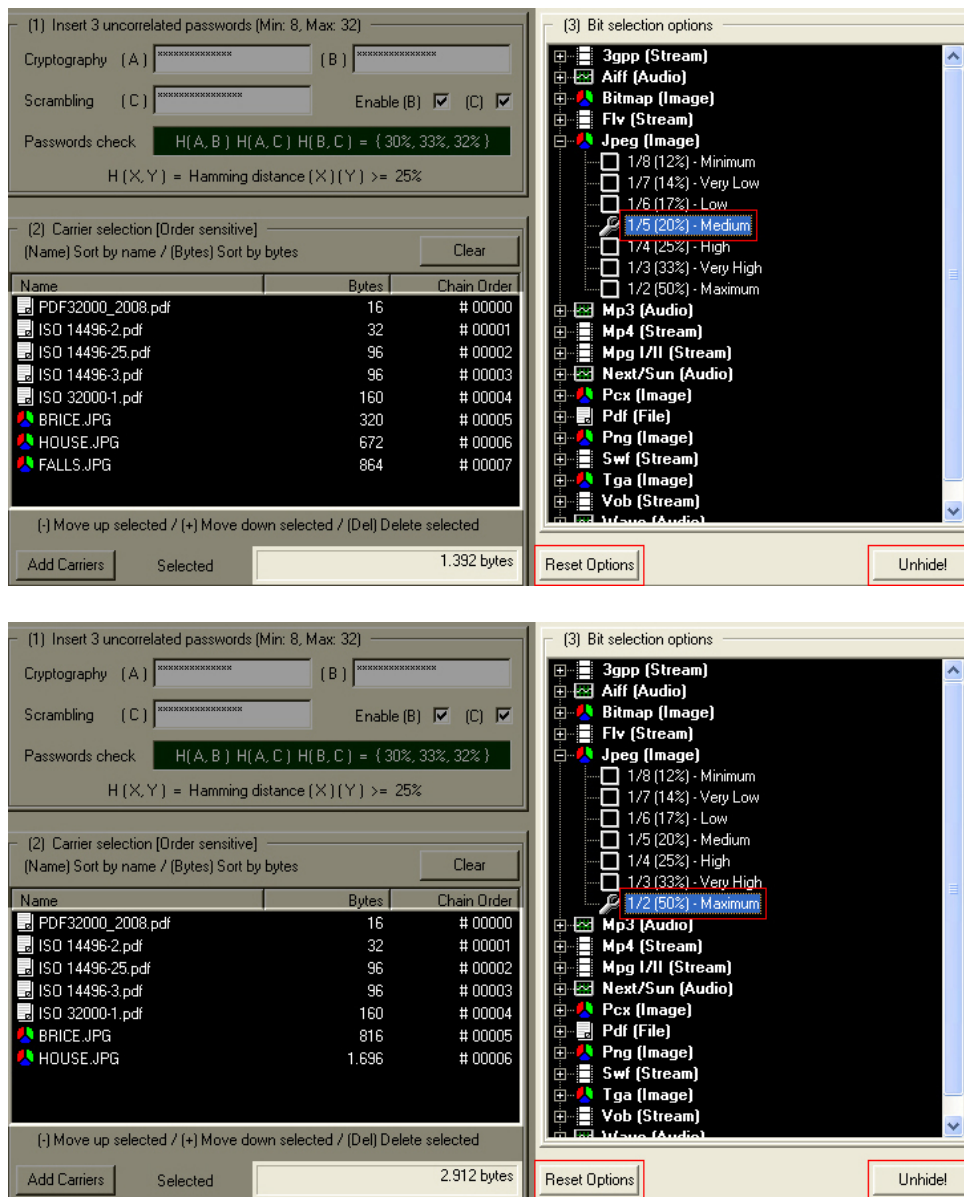
[SUGGERIMENTI PER RISULTATI MIGLIORI](#)
[COSA È LA STEGANOGRAFIA NEGABILE?](#)

PASSO 2:

(Clear)	Scartare tutti i carrier
(Add)	Aggiungere nuovi carrier alla lista
(Name)/ (Bits)	Ordinare i carrier per nome/bit
(+)/(-)	Muovere in su/giù i carrier selezionati
(Del)	Eliminare i carrier selezionati

Aggiungere tutti i carrier che sono stati processati durante l'operazione di hide.

PASSO 3:



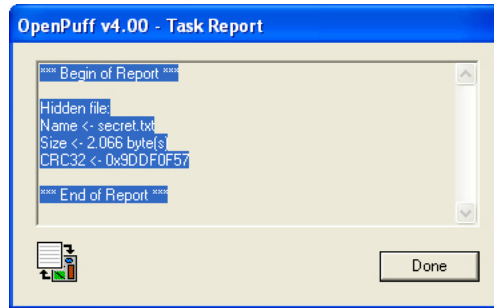
(Reset Options)	Reset dei livelli di selezione bit
(Hide!)	Inizio dell'operazione di unhide

Dopo

- aver inserito due volte la stessa password
 - aver aggiunto tutti i carrier nel giusto ordine
 - aver impostato tutti i livelli di selezione bit ai valori originali
- iniziare l'operazione di unhide

[OPZIONI: LIVELLO DI SELEZIONE BIT](#)

RAPPORTO FINALE:



Se i carrier sono stati aggiunti nel giusto ordine, con i livelli di selezione bit originali, OpenPuff sarà in grado di ricostruire i dati originali. Per una migliore sicurezza, i dati saranno ricostruiti solo dopo un controllo positivo del CRC.

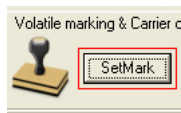
Anche la minima modifica di uno solo dei carrier potrebbe danneggiare i dati nascosti e impedire ogni tentativo di recupero.

[INDIETRO](#)



MARK SETTING IN DETTAGLIO

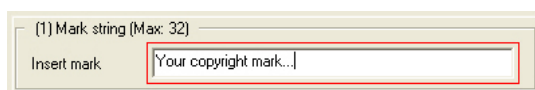
INIZIO:



(Set Mark)	Vai al pannello Mark Setting
------------------------------	------------------------------

Selezionare *Set Mark*.

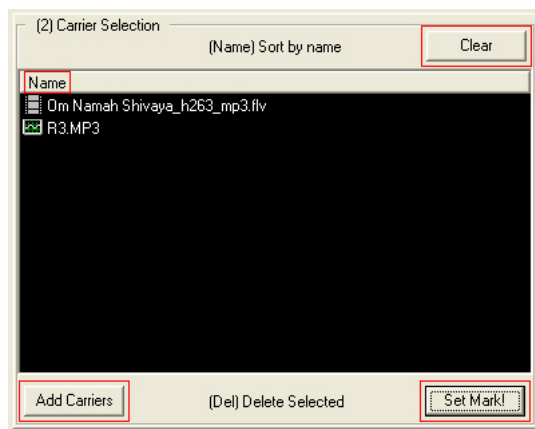
PASSO 1:



(Insert mark)	Il vostro marchio
---------------------------------	-------------------

Inserire una volta il marchio.

PASSO 2:



(Clear)	Scartare tutti i carrier
(Add)	Aggiungere nuovi carrier alla lista
(Name)	Ordinare i carrier per nome
(Del)	Eliminare i carrier selezionati
(Set Mark!)	Inizio dell'operazione di mark setting

Aggiungere tutti i carrier da marchiare.
Iniziare l'operazione di mark setting.

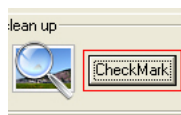
[FORMATI SUPPORTATI IN DETTAGLIO](#)

[INDIETRO](#)



MARK CHECKING IN DETTAGLIO

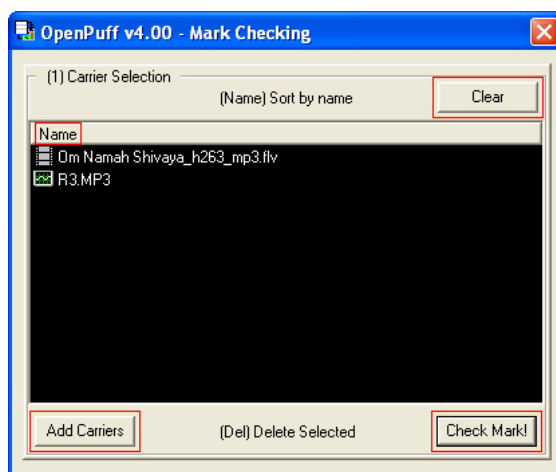
INIZIO:



(Check Mark)	Vai al pannello Mark Checking
--------------------------------	-------------------------------

Selezionare *Check Mark*.

PASSO 1:

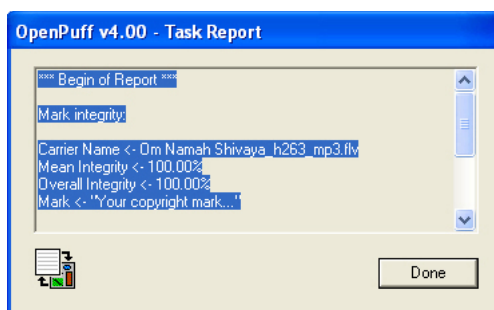


(Clear)	Scartare tutti i carrier
(Add)	Aggiungere nuovi carrier alla lista
(Name)	Ordinare i carrier per nome
(Del)	Eliminare i carrier selezionati
(Set Mark!)	Inizio dell'operazione di mark checking

Aggiungere tutti i carrier da controllare. Iniziare l'operazione di mark checking.

[FORMATI SUPPORTATI IN DETTAGLIO](#)

RAPPORTO FINALE:



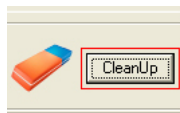
Il rapporto finale riassume, per ogni carrier, informazioni sull'integrità e sull'integrità media.

[INDIETRO](#)



DATA & MARK ERASING IN DETTAGLIO

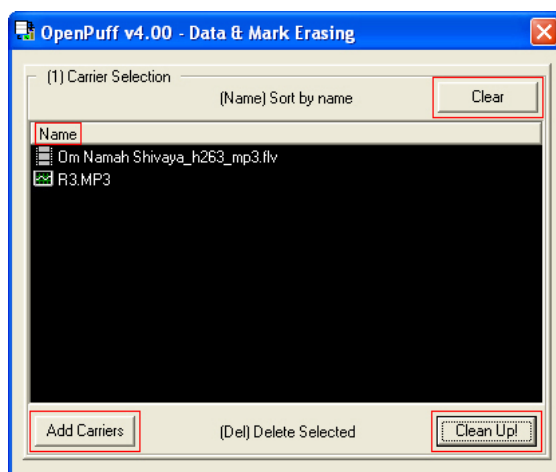
INIZIO:



(Clean Up)	Vai al pannello Data & Mark Erasing
------------	-------------------------------------

Selezionare *Clean Up*.

PASSO 1:



(Clear)	Scartare tutti i carrier
(Add)	Aggiungere nuovi carrier alla lista
(Name)	Ordinare i carrier per nome
(Del)	Eliminare i carrier selezionati
(Clean Up!)	Inizio dell'operazione di data & mark erasing

Aggiungere tutti i carrier da ripulire e iniziare l'operazione di data & mark checking.

[FORMATI SUPPORTATI IN DETTAGLIO](#)

[INDIETRO](#)