MultiObfuscator v2.00 Cryptography & Obfuscation

Advanced file & text locking made easy, safe and free

Eng. Cosimo Oliboni, Italy, 2012
Send your suggestions, comments, bug reports, requests
to *oliboni@embeddedsw.net*

MultiObfuscator HomePage

Easy	
Medium	
Expert	
Expert	

Easy	
Medium	
Expert	
Expert	

Legal remarks

Remember: this program was not written for illegal use. Usage of this program that may violate your country's laws is severely forbidden. The author declines all responsibilities for improper use of this program.

No patented code or format has been added to this program.

This is a Freeware Software

This software is released under CC BY-ND 3.0

You're free to copy, distribute, remix and make commercial use of this software under the following conditions:

- You have to cite the author (and copyright owner): *Eng. Cosimo Oliboni*
- You have to provide a link to the author's Homepage: Embeddedsw.net

Back

Features: why is this cryptography tool different from the others?

MultiObfuscator is a professional cryptography tool, with unique features you won't find among any other free or commercial software. MultiObfuscator is 100% free and suitable for highly sensitive data storage and transmission.

Let's take a look at its features

- [LAYERS OF SECURITY]
  Data is encrypted (1), scrambled (2) and whitened (3).
  FEATURES: PROGRAM ARCHITECTURE

  - [LAYER 1 - MODERN MULTI-CRIPTOGRAPHY]
    A set of 16 modern 256bit open-source cryptography algorithms has been joined into a double-password multi-cryptography algorithm (256bit+256bit).

  - [LAYER 2 - CSPRNG BASED SCRAMBLING]
    Encrypted data is always scrambled to break any remaining stream pattern. A new cryptographically secure pseudo random number generator (CSPRNG) is seeded with a third password (256bit) and data is globally shuffled with random indexes.

  - [LAYER 3 - CSPRNG BASED WHITENING]
    Scrambled data is always mixed with a high amount of noise. A new CSPRNG is seeded with a forth password (256bit) and data is bit-by-bit split according to a random permutation.

  - [EXTRA SECURITY - DENIABLE CRYPTOGRAPHY]
    Top secret data can be protected using less secret data as a decoy.
    WHAT IS DENIABLE CRYPTOGRAPHY?

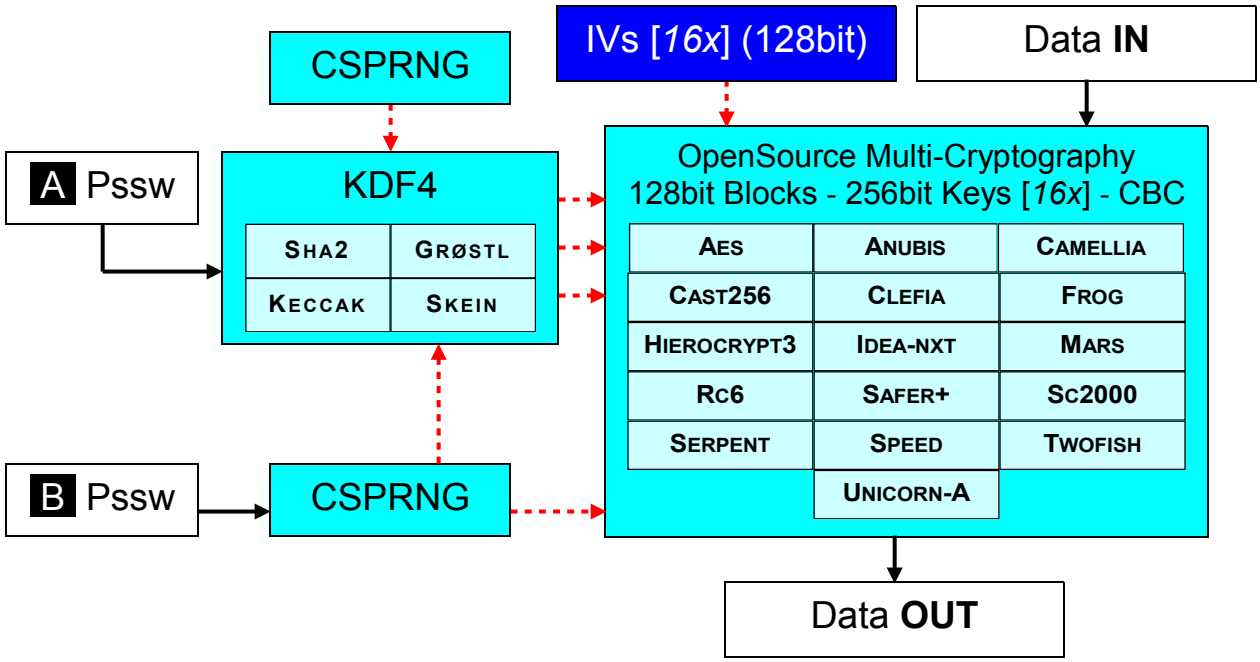
- [SOURCE CODE]
  This program can be considered as a simple Windows GUI to the LIBOBFUSCATE system-independent open-source library. Users and developers are absolutely free to link to the core library (100% of the cryptography & obfuscation code), read it and modify it.

  You're kindly asked to send me any libObfuscate porting/upgrade/customizing/derived sw, in order to analyze them and add them to the project homepage. A central updated official repository will avoid sparseness and unreachability of the project derived code.

BACK

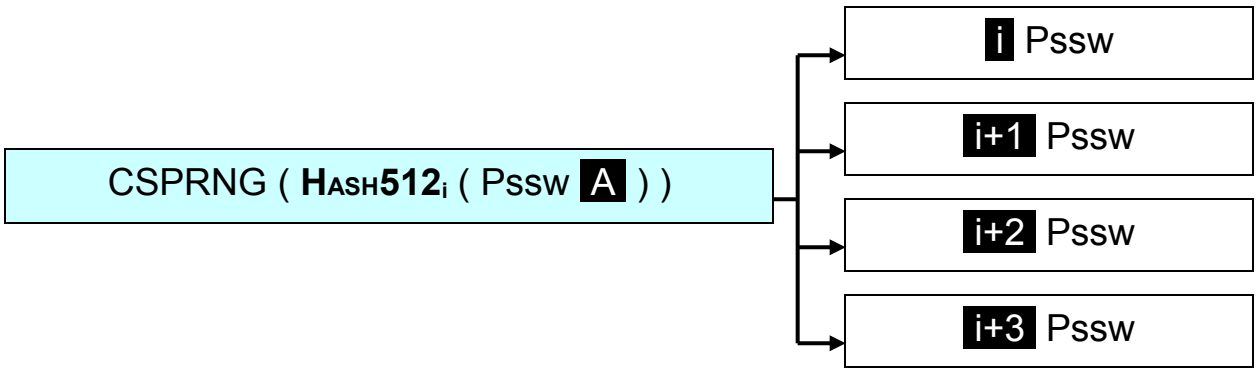
FEATURES: PROGRAM ARCHITECTURE

MultiObfuscator implements multi-cryptography (an advanced kind of PROBABILISTIC ENCRYPTION) joining 16 open-source block-based modern cryptography algorithms, chosen among AES-PROCESS, NESSIE-PROCESS and CRYPTREC-PROCESS. Cypher-Block-Chaining (CBC) wraps these block-based algorithms, letting them to behave as stream-based algorithms.



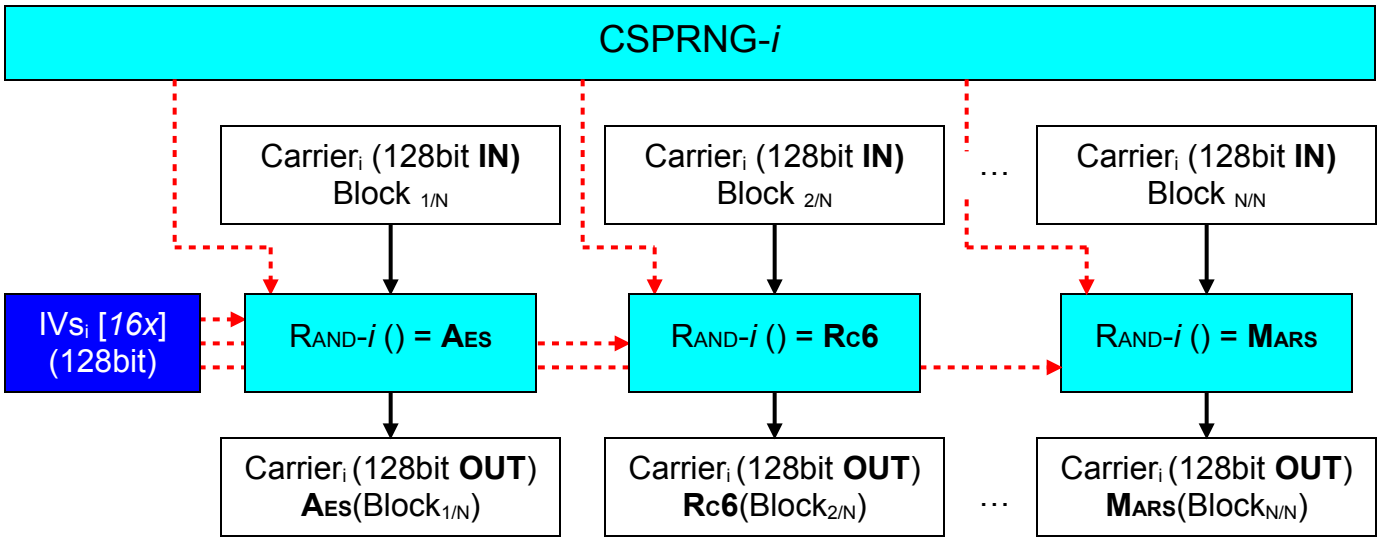
Multi-cryptography setup is a 4 step process

- a random initialization vector array (16 x 128bit) is associated to each carrier
- a pseudo random engine (CSPRNG) is seeded using password (**B**)
- password (**A**) is extended (KDF4) using 4 open-source modern 512bit hashing algorithms, taken from SHA2 and SHA3. Each hash generates four 256bit keys
  Pssw (**1**) | (**2**) | (**3**) | (**4**) = *Rand* (*Sha2* (*Pssw* (**A**)))
  Pssw (**5**) | (**6**) | (**7**) | (**8**) = *Rand* (*Grøstl* (*Pssw* (**A**)))
  Pssw (**9**) | (**10**) | (**11**) | (**12**) = *Rand* (*Keccak* (*Pssw* (**A**)))
  Pssw (**13**) | (**14**) | (**15**) | (**16**) = *Rand* (*Skein* (*Pssw* (**A**)))
- resulting key array (16 x 256bit) is associated to each cipher using the CSPRNG



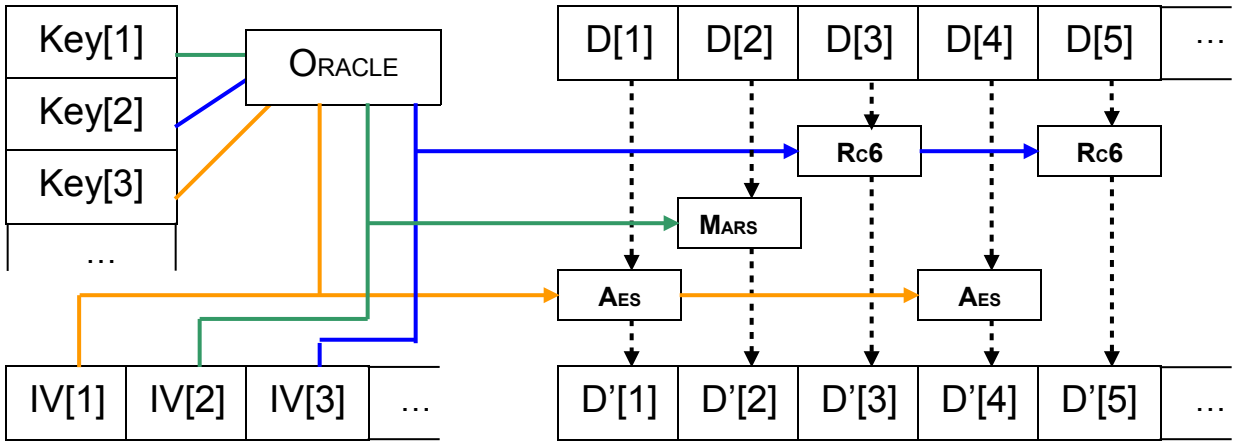
Cryptography is a multi step process

- each data gets a global setup
  Setup = { { *IV* } , *CSPRNG* , { *Key* } }
- each cipher gets an independent setup
  $Cipher_j$ = { IV_j , Key_j }
- each data block is processed with a different cipher, selected using the CSPRNG
  $CryptedBlock_k = r \leftarrow Rand\text{-}i\ ()\ ;\ Cipher_r\ (\ IV_r\ ,\ Key_r\ ,\ Block_k\ )$



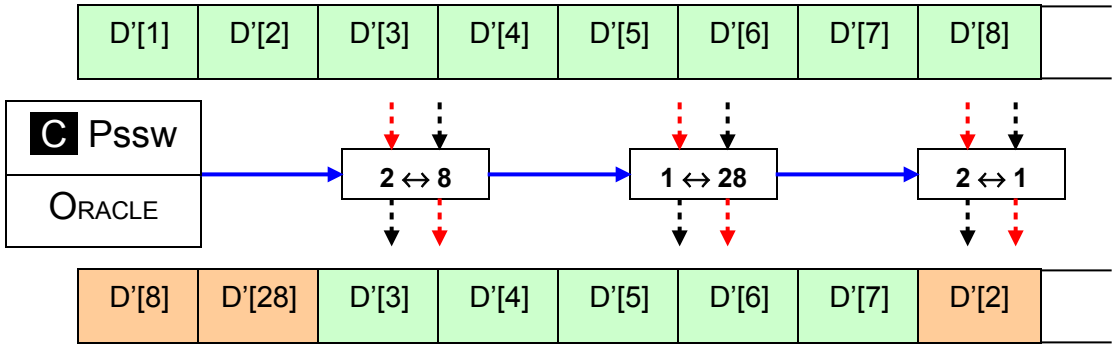
Multi-cryptography is the first layer of (local) obfuscation

- cryptography setup and CSPRNG setup get two independent passwords
- each implemented cipher gets a different IV and key
- CSPRNG behaves like an ORACLE that feeds the cryptography engine during all his choices (which key has to be associated to which cipher, which cipher has to be applied to which data block, …)



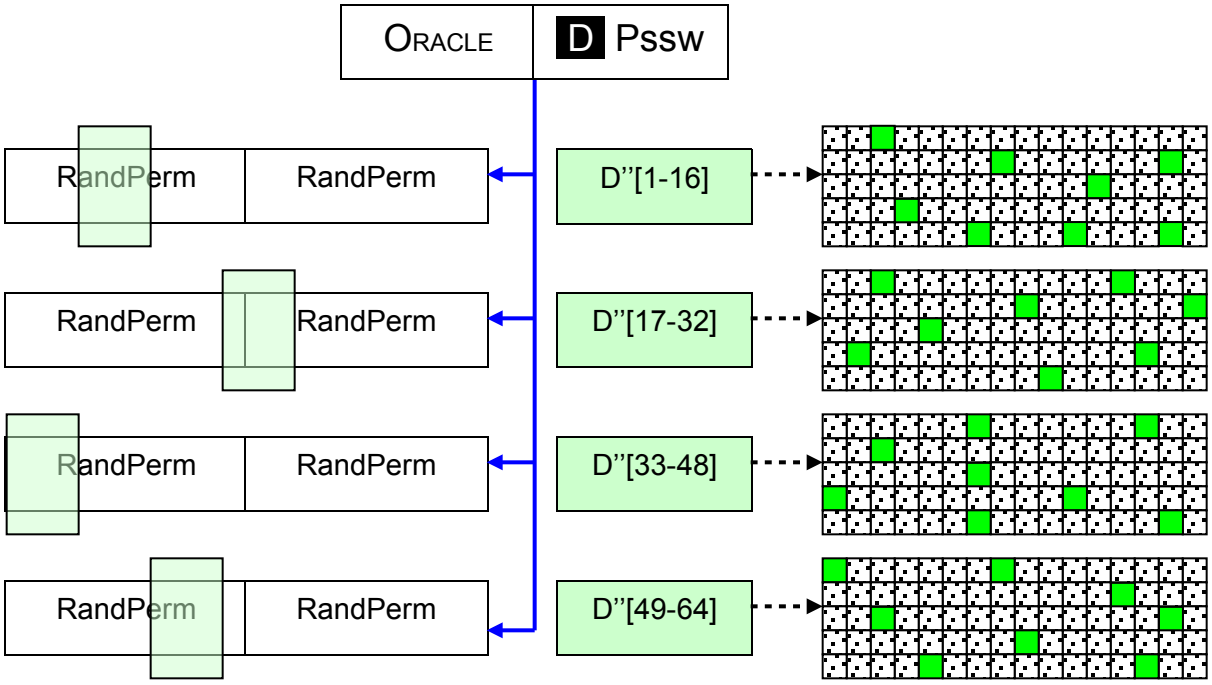
Scrambling is the second layer of (global) obfuscation

- CSPRNG setup gets an independent password
- CSPRNG behaves like an ORACLE that, given a **n**-bytes input stream, performs **n/2** random IN-PLACE shuffles, with no constraint on repeated indexes



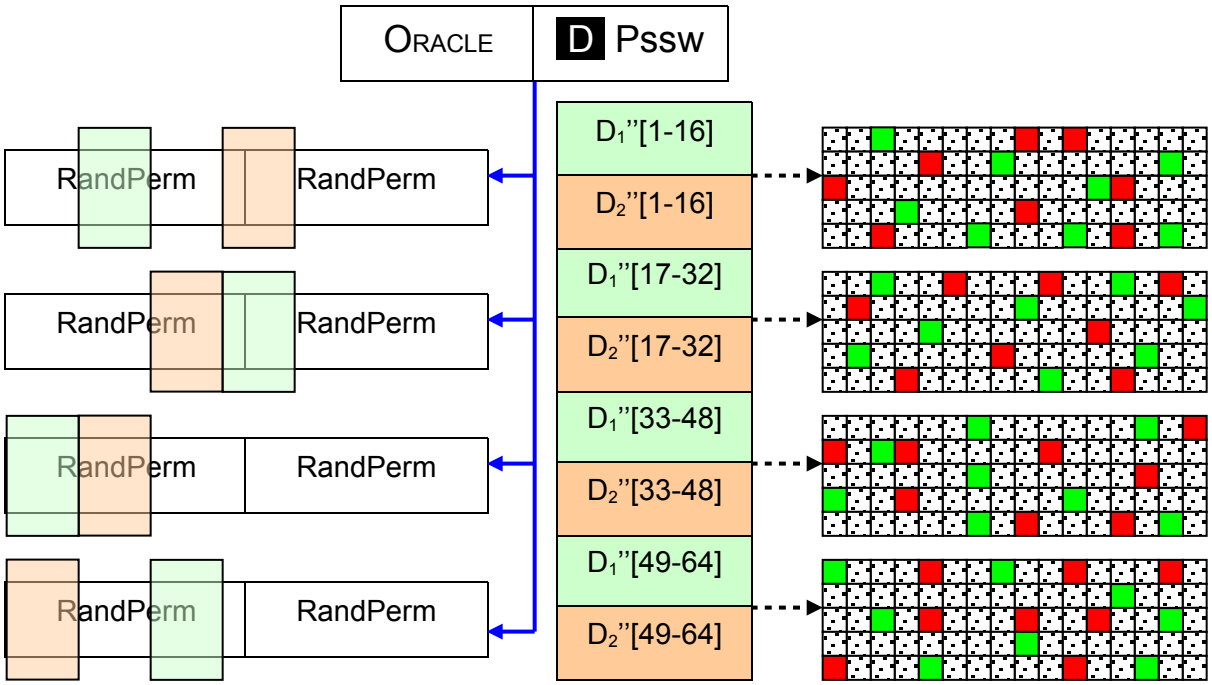
Whitening is the third layer of (local) obfuscation

- CSPRNG setup gets an independent password
- data and noise, depending on the noise level, are mixed into fixed-size blocks
  minimum: 300% noise [720 bytes] / data [240 bytes]
  maximum: 5900% noise [944 bytes] / data [16 bytes]
- 960 bytes (= 3x4x5x16) blocks force attackers to test all available noise levels (9)
- CSPRNG behaves like an ORACLE that, given a DURSTENFELD'S-SHUFFLED P-BOX (a bit-level permutation), feeds the mixing engine with a randomly slided set of non-overlapping indexes.
  OPTIONS: NOISE LEVEL



Whitening is also the core of Deniable encryption

- MultiObfuscator supports data and decoy (a 1st level of deniable encryption)
- libObfuscate supports data and many decoys (**n** levels of deniable encryption, called *aspects*)
- maximum aspects number depends, at libObfuscate core level, on the noise level
  minimum: 300% 4x aspects [240 bytes]
  maximum: 5900% 60x aspects [16 bytes]
- *Aspect* ↔ *Offset* is a random password-independent association
- *Aspect* ↔ *Offset* association, after whitening, is simply discarded
- MultiObfuscator (and any libObfuscate-linked system) is, by construction, not able to reconstruct the *Aspect* ↔ *Offset* association and, at unlocking time, has to slowly guess it by trial and error
  What is deniable cryptography?



Last OpenPuff/MultiObfuscator releases share some unique features with the Rubberhose Filesystem project (1997-2000). Independent and convergent evolution has lead different authors to focus their efforts on a common goal: Plausible deniability.

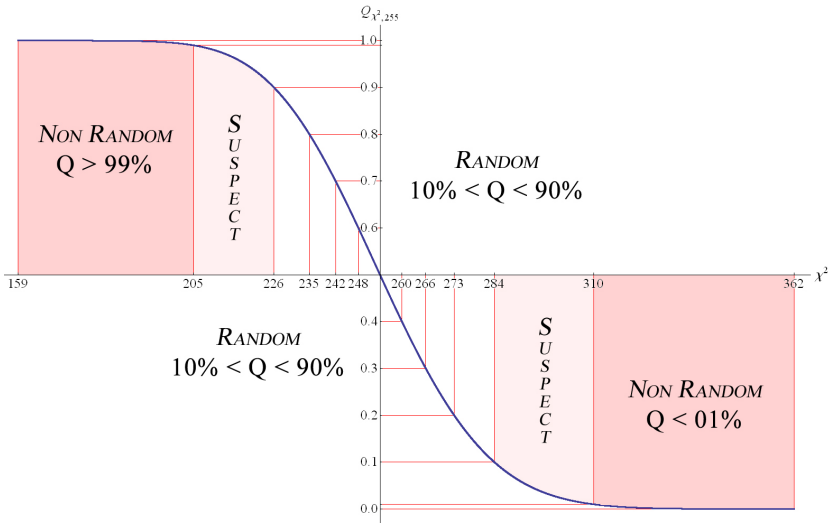
Rubberhose *was* (since it's no more maintained) a really advanced project introducing novel concepts

- **aspects**: users provide different passwords and get, from the same container, different data
- **plausible deniability**: the last-man-standing defense against legal and physical coercion

Years have gone by and, unfortunately, modern attackers wouldn't be deceived any more by whitening-only obfuscation. Batteries of statistical tests for random number generators (NIST, DieHard, ENT) would easily detect the Randomness degradation of your container and, by direct relationship, the amount of data it's been hidden inside.

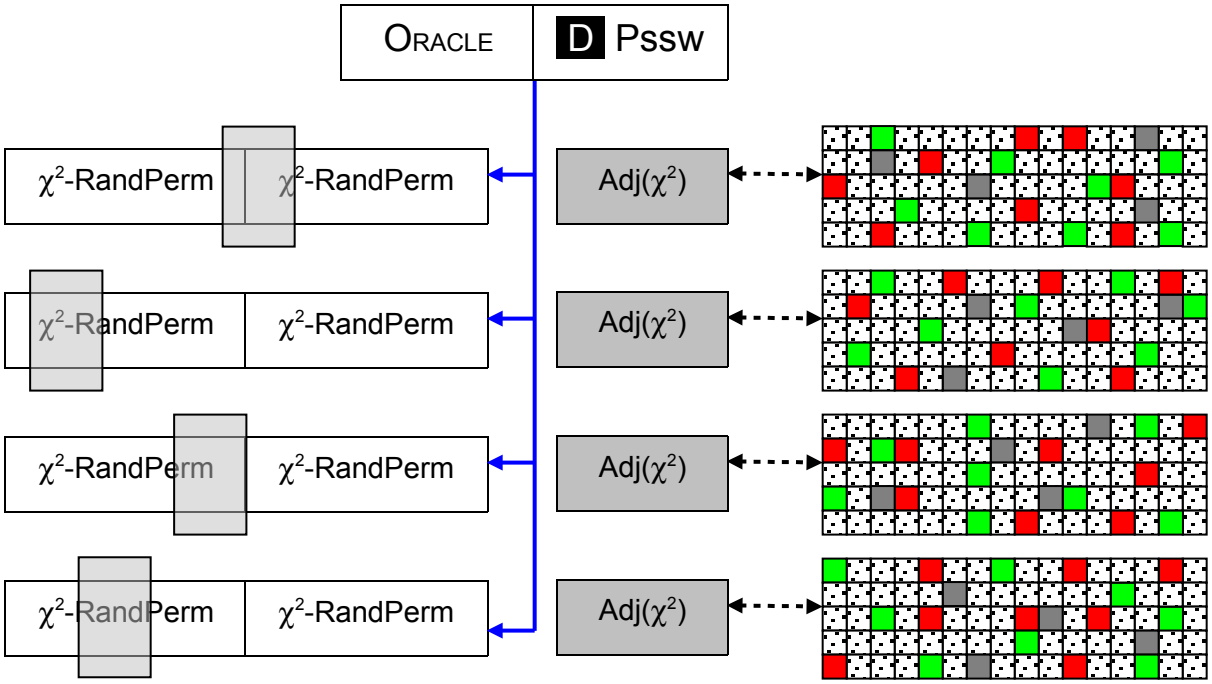
MultiObfuscator (and any libObfuscate-linked system) implements a χ^2-DISTRIBUTION-driven self-adjustment. A few bytes are randomly added into each block, letting each container, regardless of its usage (*void* → white noise, *sparse* → single aspect, *full* → **n** aspects),

- exceed χ^2-DISTRIBUTION 50% of the times (Q = 0.5), like a genuine random sequence created by RADIOACTIVE DECAY EVENTS
- score a ≥98% on the NIST randomness rating system



Advanced users will take great advantage of statistical resitant containers

- adding void/fake containers to the sensitive ones, in order to waste attackers' time
- always convincingly denying you're using more than a single aspect
- sharing, unbeknownst to everybody, a multi-aspect container among untrusted, potentially malicious, people



BACK

FAQ 1: Why didn't you simply implement a standard AES-256 or RSA-1024?

Modern open-source cryptography
- has been thoroughly investigated and reviewed by the scientific community
- it's widely accepted as the safest way to secure your data
- fulfills almost every *standard* need of security

MultiObfuscator doesn't support any CONSPIRACY THEORY against our privacy (SECRET CRACKING BACKDOORS, intentionally weak cryptography designs, …). There's really no reason not to trust standard modern publicly available cryptography (although some old ciphers have been already CRACKED).

Some users, however, are very likely to be hiding very sensitive data, with an *unusually high* need of security. Their secrets need to go through a deep process of data OBFUSCATION in order to be able to *longer* survive forensic investigation and hardware aided brute force attacks.

FAQ 2: Is multi-cryptography similar to multiple-encryption?

Multi-cryptography is something really different from MULTIPLE-ENCRYPTION (encrypting more than once). There's really no common agreement about multiple-encryption's reliability. It's thought to be:
- BETTER than single encryption
- WEAK as the weakest cipher in the encryption queue/process
- **worse** than single encryption

MultiObfuscator supports the last thesis (worse) and never encrypts already encrypted data.

FAQ 3: Is multi-cryptography similar to random/polymorpihc-cryptography?

Random-cryptography, a.k.a. POLYMORPHIC CRYPTOGRAPHY, is a well-known SNAKE-OIL CRYPTOGRAPHY. Multi-cryptography is something completely different and never aims to build some better, random or on-the-fly cipher.

MultiObfuscator only relies on stable modern opern-source cryptography.

FAQ 4: Is multi-cryptography better than standard cryptography?

Is a *house* better than a *brick*? No. The house is a *superstructure* and a brick is a *material*.

Is *multi-cryptography* better than *cryptography*? No. Multi-cryptography is part of a data obfuscation *process* and cryptography is a *component*.

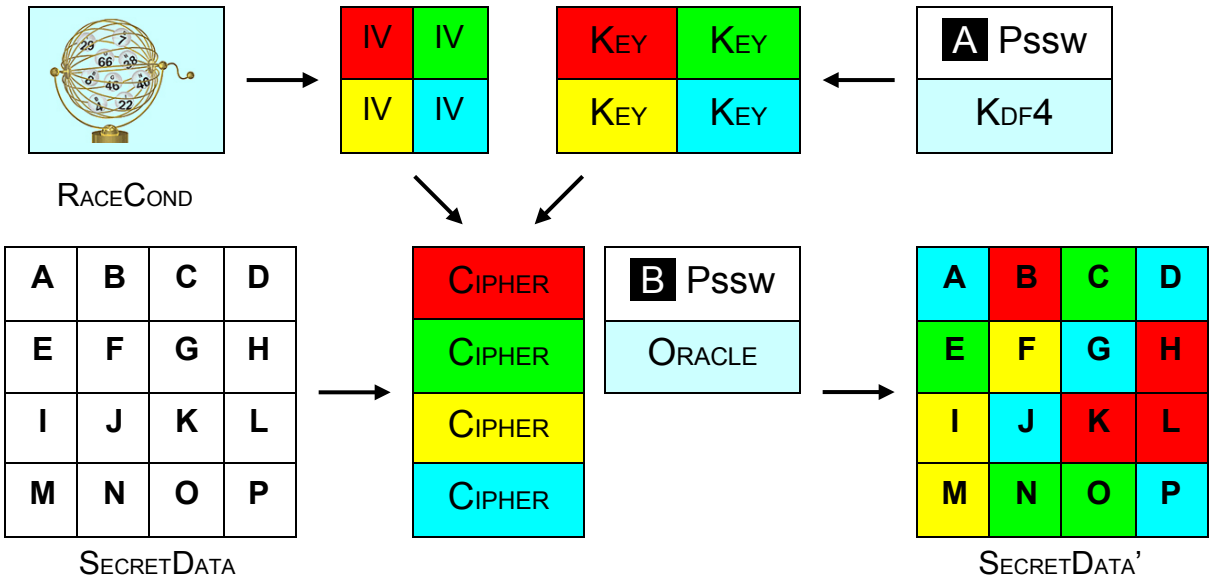
Cryptography	=	Brick
Multi-cryptography	=	Floor
Obfuscation process	=	House

FAQ 5: Is data obfuscation better than standard cryptography?

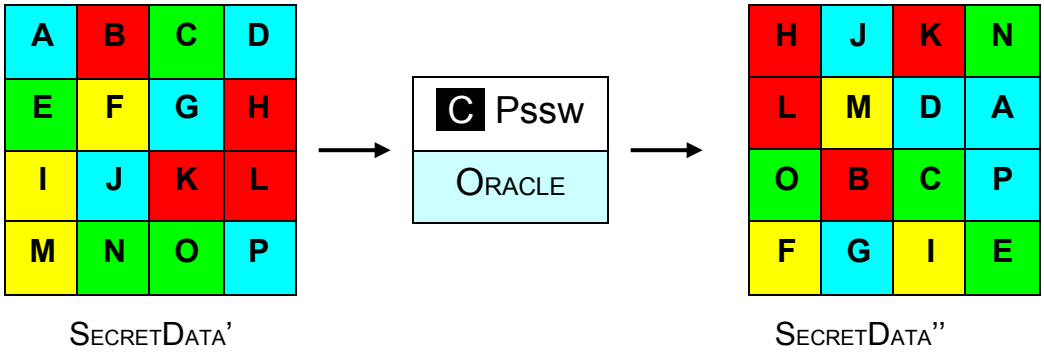
Data obfuscation never raises any claim of UNBREAKABILITY (always to be considered as a symptom of fake software or SNAKE OIL CRYPTOGRAPHY). Yet it's still possible to handle the "*unusually high need of security*"-problem in an effective and constructive way (according to KERCKHOFF'S PRINCIPLE), as an engineering task (*slowing attackers down* as much as possible)

- connecting different obfuscation transformations
- avoiding repeatedly applying the same transformation
- relying only on open-source resources
- applying some global transformation, software-only reversible

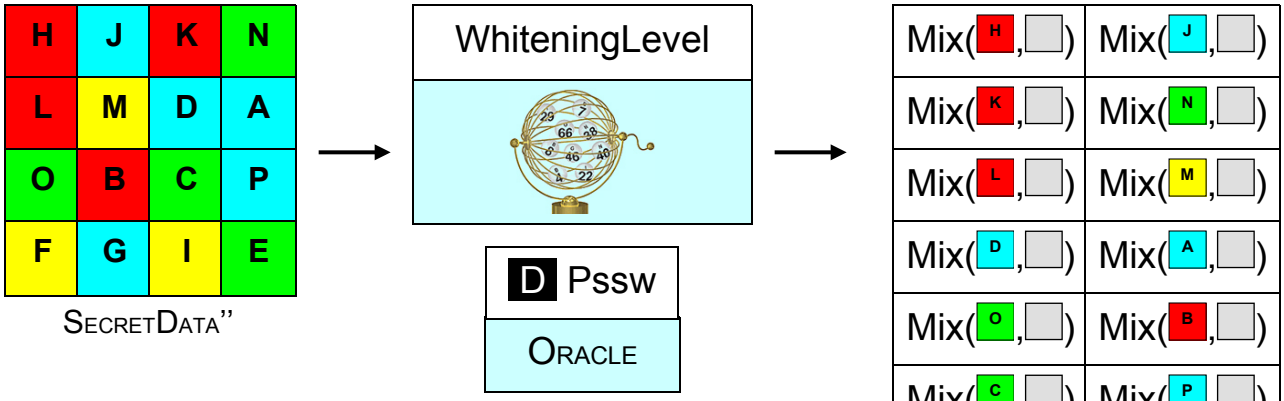
[Round 1 – MULTI-CRYPTOGRAPHY (LOCAL TRANSFORMATION)]



[Round 2 – SCRAMBLING (GLOBAL TRANSFORMATION)]



[Round 3 – WHITENING (LOCAL TRANSFORMATION)]



SECRETDATA''

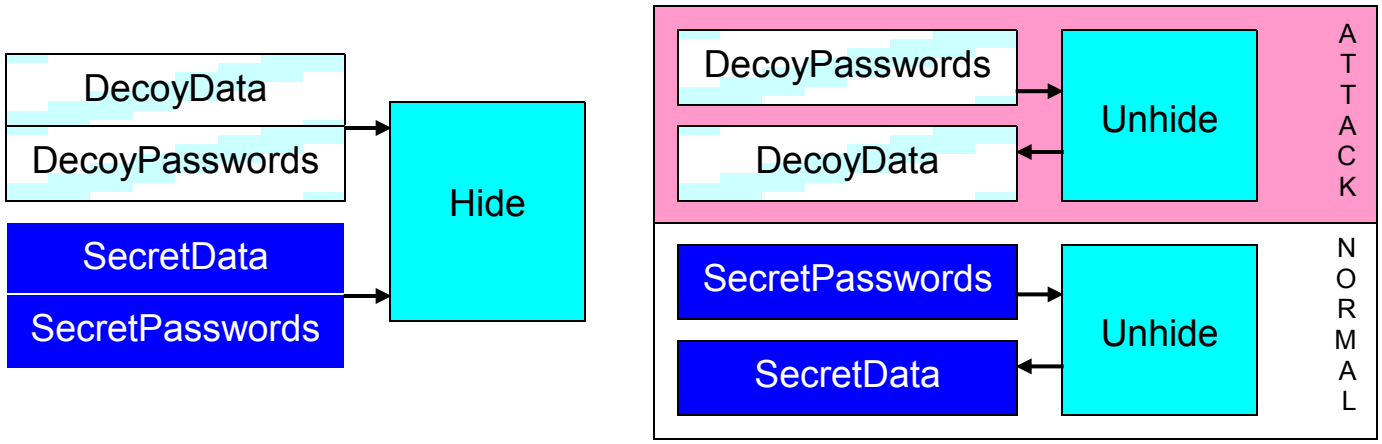
SECRETDATA'''

FEATURES: PROGRAM ARCHITECTURE

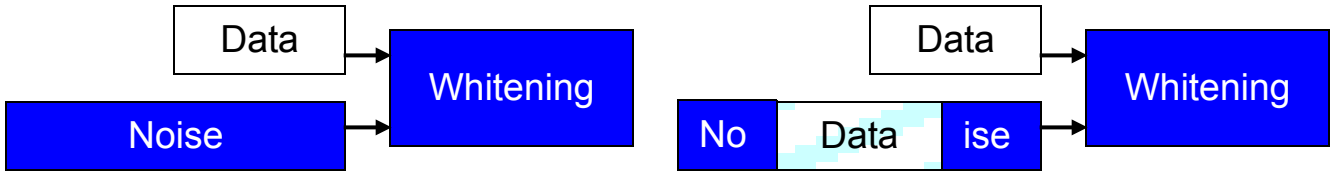
BACK

What is deniable cryptography?

Deniable encryption is a decoy based technique that allows you to convincingly deny the fact that you're hiding **sensitive data**, even if attackers are able to state that you're hiding some data. You only have to provide some expendable decoy data that you would Plausibly want to keep confidential. It will be revealed to the attacker, claiming that this is all there is.



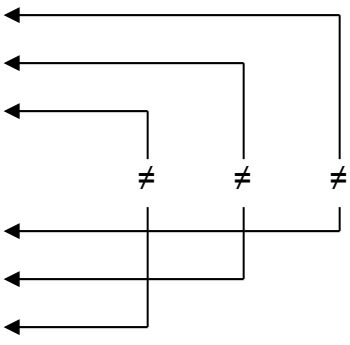
How is it possible? Encrypted and scrambled data is whitened (Features: program architecture) with a high amount of noise. Decoy data can replace some of this noise without loosing final properties of Cryptanalysis resistance.



Sensitive data and decoy data are encrypted using different passwords. You have to choose two different sets of different passwords.

Example:

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%

Each password has to be different (at bit level) and at least 8 characters long.

Example: “DataPssw1” (A) “DataPssw2” (B) “DataPssw3” (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**

Example: “FirstDataPssw1” (A) “SecondDataPssw2” (B) “AnotherDataPssw3” (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**

You will be asked for

- two **different** sets of different passwords
- a stream of sensitive data
- a stream of decoy data **compatible** (by size) with sensitive data
  $\sum_{k \in \{1, N-1\}} used\_bytes(whiteBlock_k) < Sizeof(Decoy) \leq \sum_{k \in \{1, N\}} used\_bytes(whiteBlock_k)$

Example:

whiteBlocks	Data bytes	SensitiveData	DecoyData
+Block (1/N)	32	32	Used
…	2016	2016	Used
+Block (N-1/N)	32	32	Used
+Block (N/N)	32	15	1 – 32
	Total = 2112	Total = 2095	2080 < Size ≤ 2112

BACK

Options: noise level

File mode:

- **Format**: raw binary file
- **Fixed size block**: Noise + Data = 960 bytes
- **Locked output size**: ((size + 256) / Data) * 960 ≤ 256 Mb

Noise Level	Noise	Data	Min. Plain → Locked Size	Max. Plain → Locked Size
300%	720	**240**	1 B → 1920 B	64 Mb → 256 Mb
				
400%	768	**192**	1 B → 1920 B	51 Mb → 256 Mb
				
500%	800	**160**	1 B → 1920 B	42 Mb → 256 Mb
				
900%	864	**96**	1 B → 2880 B	25 Mb → 256 Mb
				
1100%	880	**80**	1 B → 3840 B	21 Mb → 256 Mb
				
1400%	896	**64**	1 B → 4800 B	17 Mb → 256 Mb
				
1900%	912	**48**	1 B → 5760 B	12 Mb → 256 Mb
				
2900%	928	**32**	1 B → 8640 B	8 Mb → 256 Mb
				
5900%	944	**16**	1 B → 16320 B	4 Mb → 256 Mb
				

Text mode:

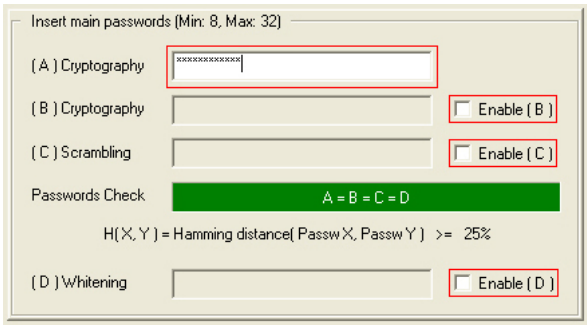
- **Format**: text/email
- **Fixed size block**: Noise + Data = 960 bytes → 6 bit encoding → 1280 bytes
- **Locked output size**: ((size + 256) / Data) * 1280 ≤ 256 Kb

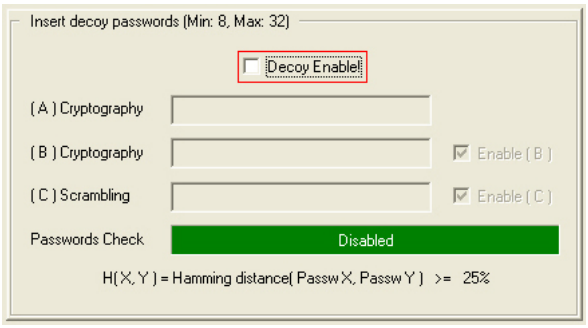
Noise Level	Noise	Data	Min. Plain → Locked Size	Max. Plain → Locked Size
300%	720	**240**	1 B → 2560 B	46 Kb → 256 Kb
	Whitening 300%: 720 noise / 240 data			
400%	768	**192**	1 B → 2560 B	36 Kb → 256 Kb
	Whitening 400%: 768 noise / 192 data			
500%	800	**160**	1 B → 2560 B	30 Kb → 256 Kb
	Whitening 500%: 800 noise / 160 data			
900%	864	**96**	1 B → 3840 B	18 Kb → 256 Kb
	Whitening 900%: 864 noise / 96 data			
1100%	880	**80**	1 B → 5120 B	15 Kb → 256 Kb
	Whitening 1100%: 880 noise / 80 data			
1400%	896	**64**	1 B → 6400 B	12 Kb → 256 Kb
	Whitening 1400%: 896 noise / 64 data			
1900%	912	**48**	1 B → 7680 B	9 Kb → 256 Kb
	Whitening 1900%: 912 noise / 48 data			
2900%	928	**32**	1 B → 11520 B	6 Kb → 256 Kb
	Whitening 2900%: 928 noise / 32 data			
5900%	944	**16**	1 B → 21760 B	3 Kb → 256 Kb
	Whitening 5900%: 944 noise / 16 data			

BACK

EASY PASSWORDS SETUP

EASY FILE/TEXT LOCK/UNLOCK – BASE SETUP (1 PASSWORD)

(I) 

(II) 

(I)	(*Cryptography A*)	First password
	(*Enable B*)	Second password enable/disable
	(*Enable C*)	Third password enable/disable
	(*Enable D*)	Forth password enable/disable
(II)	(*Decoy Enable!*)	Decoy enable/disable

A) Disable decoy

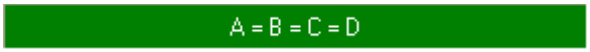
B.1) Disable all optional (Main_B / Main_C / Main_D) passwords
B.2) Enter any (Main_A) password

Disabled (Main_B / Main_C / Main_D) passwords will be set same as (Main_A) password!

Constraints:

1) Length (Main_A) ≥ 8

Example:



Main: ok

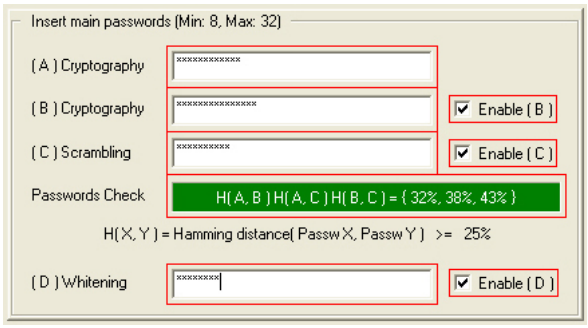
Main_A = “any password”

BACK

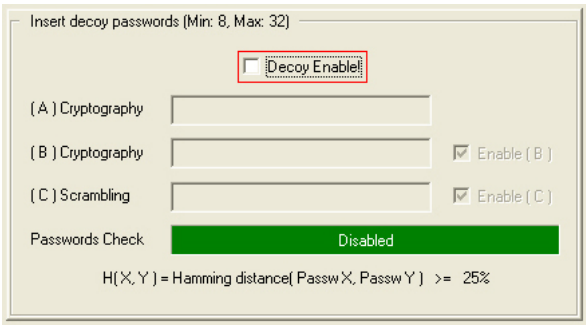
MEDIUM PASSWORDS SETUP

MEDIUM — FILE/TEXT LOCK/UNLOCK – MEDIUM SETUP (4 PASSWORDS)

(I)



(II)



(I)	(*Cryptography A*)	First password
	(*Cryptography B*)	Second password (cryptography CSPRNG)
	(*Scrambling C*)	Third password (scrambling CSPRNG)
	(*Whitening D*)	Forth password (whitening CSPRNG)
	(*Enable B*)	Second password enable/disable
	(*Enable C*)	Third password enable/disable
	(*Enable D*)	Forth password enable/disable
(II)	(*Decoy Enable!*)	Decoy enable/disable

A) Disable decoy

B.1) Enable all or only some of (Main_B / Main_C / Main_D) optional passwords
B.2) Enter different (Main_A / Main_B / Main_C) passwords
B.3) Enter any (Main_D) password

Disabled (Main_B / Main_C / Main_D) passwords will be set same as (Main_A) password!

Constraints:

1.1) Length (Main_A) ≥ 8
1.2) Enabled? (Main_B) → Length (Main_B) ≥ 8
1.3) Enabled? (Main_C) → Length (Main_C) ≥ 8
1.4) Enabled? (Main_D) → Length (Main_D) ≥ 8

2.1) Enabled? (Main_B) → HAMMING DISTANCE (Main_A / Main_B) ≥ 25%
2.2) Enabled? (Main_C) → HAMMING DISTANCE (Main_A / Main_C) ≥ 25%
2.3) Enabled? (Main_B / Main_C) → HAMMING DISTANCE (Main_B / Main_C) ≥ 25%

Example:

Main: Main_A too similar to Main_B

Main_A = “**some_crypt_a**”
Main_B = “**some_crypt_b**”
Main_C = “scramble_c”
Main_D = “whiten_d”

Main: Main_A too similar to Main_C

Main_A = “**some_crypt_a**”
Main_B = “another_crypt_b”
Main_C = “**some_crypt_c**”
Main_D = “whiten_d”

Main: Main_B too similar to Main_C

Main_A = “some_crypt_a”
Main_B = “**another_crypt_b**”
Main_C = “**another_crypt_c**”
Main_D = “whiten_d”

Main: ok

Main_A = “some_crypt_a”
Main_B = “another_crypt_b”
Main_C = “scramble_c”
Main_D = “whiten_d”

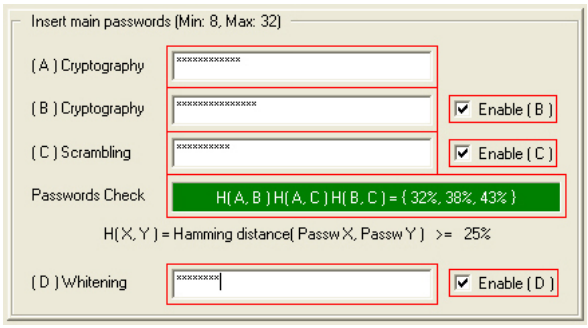
BACK

ADVANCED PASSWORDS SETUP – LOCK

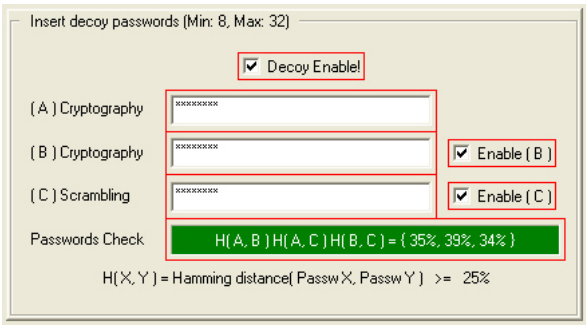
EXPERT

FILE/TEXT LOCK – ADVANCED SETUP (4 PASSWORDS+DECOY)

(I)



(II)



(I)	(*Cryptography A*)	First password
	(*Cryptography B*)	Second password (cryptography CSPRNG)
	(*Scrambling C*)	Third password (scrambling CSPRNG)
	(*Whitening D*)	Forth password (whitening CSPRNG)
	(*Enable B*)	Second password enable/disable
	(*Enable C*)	Third password enable/disable
	(*Enable D*)	Forth password enable/disable
(II)	(*Decoy Enable!*)	Decoy enable/disable
	(*Cryptography A*)	First decoy password
	(*Cryptography B*)	Second decoy password
	(*Scrambling C*)	Third decoy password
	(*Enable B*)	Second decoy password enable/disable
	(*Enable C*)	Third decoy password enable/disable

A) Disable decoy

B.1) Enable all or only some of (Main_B / Main_C / Main_D) passwords
B.2) Enter different (Main_A / Main_B / Main_C) passwords
B.3) Enter any (Main_D) password

Disabled (Main_B / Main_C / Main_D) passwords will be set same as (Main_A) password!

C) Enable decoy

D.1) Enable both or only one of (Decoy_B / Decoy_C) passwords
D.2) Enter different (Decoy_A / Decoy_B / Decoy_C) passwords

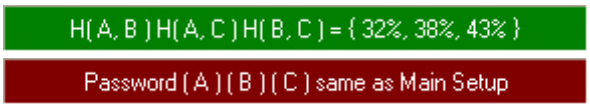
Disabled (Decoy_B / Decoy_C) passwords will be set same as (Decoy_A) password!

Constraints:

1.1)			Length (Main_A) ≥ 8
1.2)	Enabled? (Main_B)	→	Length (Main_B) ≥ 8
1.3)	Enabled? (Main_C)	→	Length (Main_C) ≥ 8
1.4)	Enabled? (Main_D)	→	Length (Main_D) ≥ 8
2.1)	Enabled? (Main_B)	→	HAMMING DISTANCE (Main_A / Main_B) ≥ 25%
2.2)	Enabled? (Main_C)	→	HAMMING DISTANCE (Main_A / Main_C) ≥ 25%
2.3)	Enabled? (Main_B / Main_C)	→	HAMMING DISTANCE (Main_B / Main_C) ≥ 25%
3.1)			Length (Decoy_A) ≥ 8
3.2)	Enabled? (Decoy_B)	→	Length (Decoy_B) ≥ 8
3.3)	Enabled? (Decoy_C)	→	Length (Decoy_C) ≥ 8
4.1)	Enabled? (Decoy_B)	→	HAMMING DISTANCE (Decoy_A / Decoy_B) ≥ 25%
4.2)	Enabled? (Decoy_C)	→	HAMMING DISTANCE (Decoy_A / Decoy_C) ≥ 25%
4.3)	Enabled? (Decoy_B / Decoy_C)	→	HAMMING DISTANCE (Decoy_B / Decoy_C) ≥ 25%

5.1)	Enabled? (Decoy_B)	→ Enabled? (Main_B)	→ Main_B ≠ Decoy_B
5.2)	Enabled? (Decoy_B)	→ Disabled? (Main_B)	→ Main_A ≠ Decoy_B
5.3)	Enabled? (Decoy_C)	→ Enabled? (Main_C)	→ Main_C ≠ Decoy_C
5.4)	Enabled? (Decoy_C)	→ Disabled? (Main_C)	→ Main_A ≠ Decoy_C

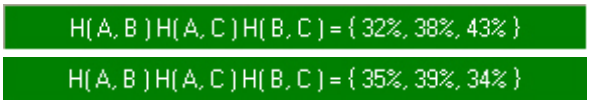
Example:



Main: ok

Decoy: Main_A = Decoy_A, …

Main_A = “some_crypt_a” Decoy_A = “**some_crypt_a**”
Main_B = “another_crypt_b” Decoy_B = “**another_crypt_b**”
Main_C = “scramble_c” Decoy_C = “**scramble_c**”
Main_D = “whiten_d”



Main: ok

Decoy: Main_A = Decoy_A, …

Main_A = “some_crypt_a” Decoy_A = “12345678”
Main_B = “another_crypt_b” Decoy_B = “qwertyui”
Main_C = “scramble_c” Decoy_C = “zxcvbnm,”
Main_D = “whiten_d”

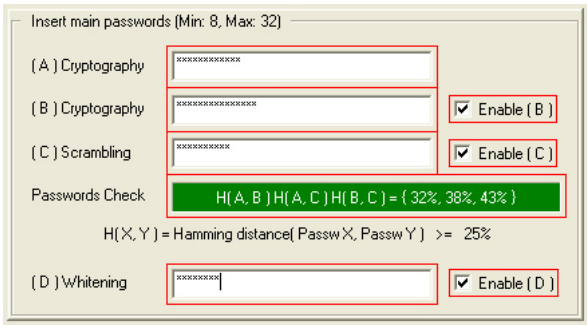
BACK

ADVANCED PASSWORDS SETUP – UNLOCK

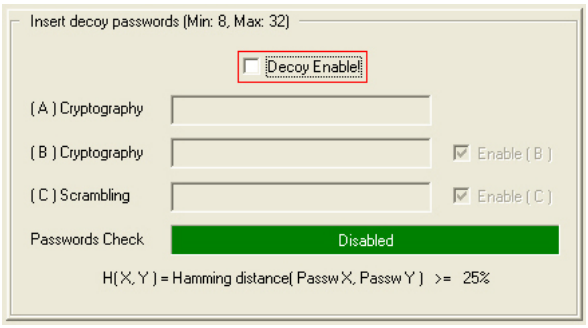
EXPERT

FILE/TEXT UNLOCK – ADVANCED SETUP (4 PASSWORDS+DECOY)

(I)



(II)



(I)	(*Cryptography A*)	First password
	(*Cryptography B*)	Second password (cryptography CSPRNG)
	(*Scrambling C*)	Third password (scrambling CSPRNG)
	(*Whitening D*)	Forth password (whitening CSPRNG)
	(*Enable B*)	Second password enable/disable
	(*Enable C*)	Third password enable/disable
	(*Enable D*)	Forth password enable/disable
(II)	(*Decoy Enable!*)	Decoy enable/disable

Example:

Lock	
Main_A = “some_crypt_a” Main_B = “another_crypt_b” Main_C = “scramble_c” Main_D = “**whiten_d**”	Decoy_A = “12345678” Decoy_B = “qwertyui” Decoy_C = “zxcvbnm,”
Secret data unlock	
Main_A = “some_crypt_a” Main_B = “another_crypt_b” Main_C = “scramble_c” Main_D = “**whiten_d**”	DISABLED
Decoy data unlock	
Main_A = “12345678” Main_B = “qwertyui” Main_C = “zxcvbnm,” Main_D = “**whiten_d**”	DISABLED

OK *Main_D password is always shared by main and decoy data*

Lock	
Main_A = “some_crypt_a” Main_B = **DISABLED** Main_C = “scramble_c” Main_D = “whiten_d”	Decoy_A = “12345678” Decoy_B = “qwertyui” Decoy_C = **DISABLED**
Secret data unlock	
Main_A = “some_crypt_a” Main_B = **DISABLED** Main_C = “scramble_c” Main_D = “whiten_d”	DISABLED
Decoy data unlock	
Main_A = “12345678” Main_B = “qwertyui” Main_C = **DISABLED** Main_D = “whiten_d”	DISABLED

OK *Main_B / Main_C / Decoy_B / Decoy_C passwords can be independently disabled*

Lock	
Main_A = “some_crypt_a” Main_B = DISABLED Main_C = “scramble_c” Main_D = **DISABLED**	Decoy_A = “12345678” Decoy_B = “qwertyui” Decoy_C = DISABLED
Secret data unlock	
Main_A = “some_crypt_a” Main_B = DISABLED Main_C = “scramble_c” Main_D = DISABLED	DISABLED
Decoy data unlock	
Main_A = “12345678” Main_B = “qwertyui” Main_C = DISABLED Main_D = “**some_crypt_a**”	DISABLED

This is a WRONG configuration:

- disabled Main_D password is set same as Main_A password
- decoy unlocking (when you’re under attack…) will reveal Main_A password to the attacker!

Never disable Main_D password if you’re planning to use a decoy.

BACK

FILE LOCK – BASE SETUP (1 PASSWORD)

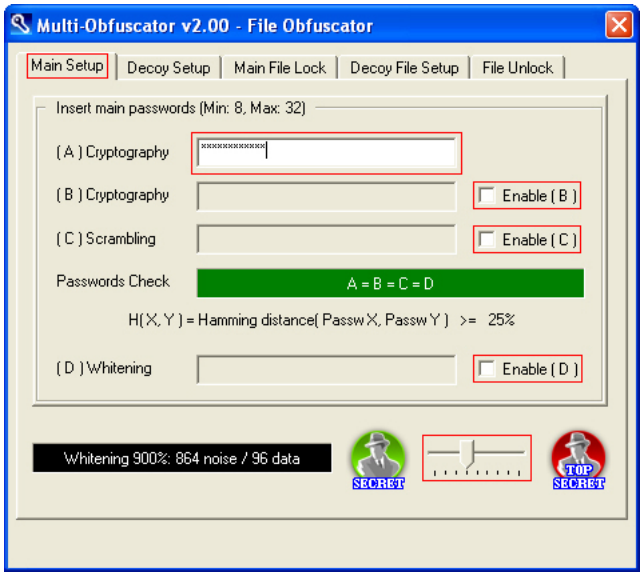
BEGIN:

(*File Lock/Unlock*)	Go to file (binary raw format) panel

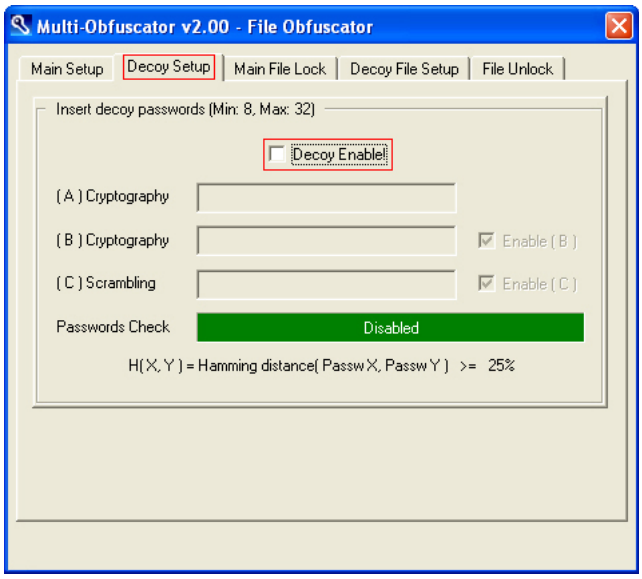
Select *File Lock/Unlock*.

STEP 1:

(I)



(II)



(I)	(*Cryptography A*)	First password
	(*Enable B*)	Second password enable/disable
	(*Enable C*)	Third password enable/disable
	(*Enable D*)	Forth password enable/disable
(II)	(*Decoy Enable!*)	Decoy enable/disable

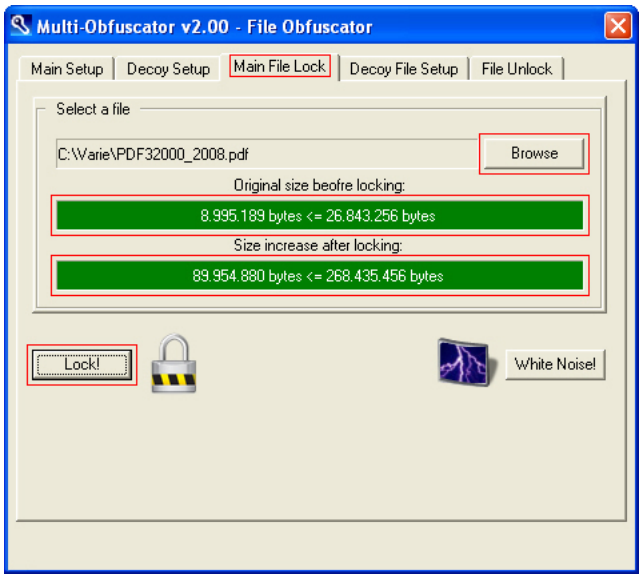
Insert a password and choose a noise level. Full password and noise details are available in special separate sections:

- EASY PASSWORDS SETUP
- OPTIONS: NOISE LEVEL

Base setup, even though looking like a traditional security software, relies on the same multi-layered security architecture as advanced setup.

FEATURES: PROGRAM ARCHITECTURE

STEP 2:



(*Browse*)	Select a file
(*Original size before locking*)	Example: 8.995.189 bytes
(*Size increase after locking*)	Example: 89.954.880 bytes
(*Lock!*)	Start locking

Choose the secret data you want to lock (a single file or a zip/rar/… archive). Secret data will not be overwritten and locked data will be saved to a different folder. File/archive name will not be saved to the locked data, allowing renaming and unlocking secret data with a different name.

Example:

- MultiObfuscator: C:\…\dir1\xxx.pdf [9 Mb] → C:\...\dir2\xxx.pdf [90 Mb]
- Rename: C:\...\dir2\xxx.pdf → UsbKey:\...\yyy.pdf
- MultiObfuscator: UsbKey:\…\yyy.pdf [90 Mb] → D:\…\yyy.pdf [9 Mb]

There's a maximum locked size constraint of 256 Mb and, depending on the noise level, there's also a maximum plain size constraint. Little files (up to 4 Mb) will let you free to choose any noise level. Medium and large files (up to 64 Mb) will force you to choose a lower compatible (by size) noise level.

Example:

- Noise level: 900%
- Original size before locking: 8.995.189 bytes ≤ 25 Mb
- Size after locking: ((8.995.189 + 256) / 96) * 960 = 89.954.880 bytes ≤ 256 Mb

Noise Level	Noise	Data	Min. Plain → Locked Size	Max. Plain → Locked Size
900%	864	**96**	1 B → 2880 B	**25 Mb → 256 Mb**

OPTIONS: NOISE LEVEL

BACK

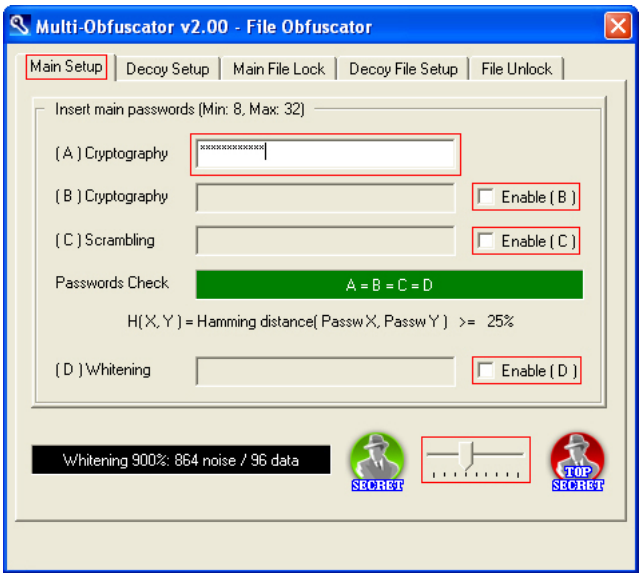
FILE UNLOCK – BASE SETUP (1 PASSWORD)

BEGIN:

(*File Lock/Unlock*)	Go to file (binary raw format) panel

Select *File Lock/Unlock*.

STEP 1:

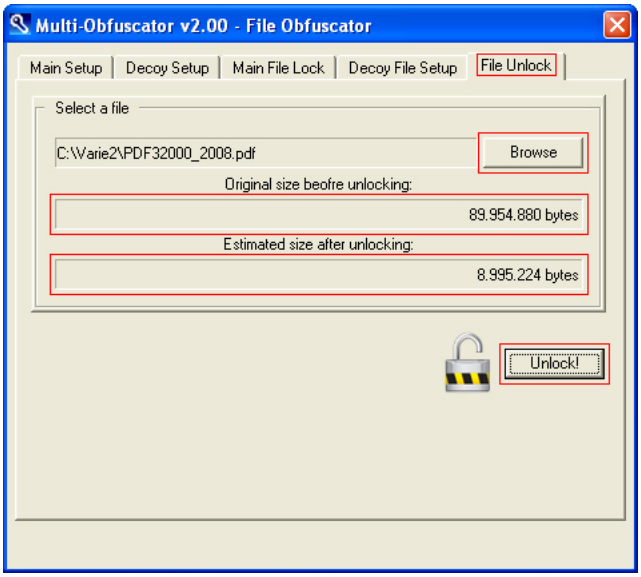


(*Cryptography A*)	First password
(*Enable B*)	Second password enable/disable
(*Enable C*)	Third password enable/disable
(*Enable D*)	Forth password enable/disable

Set same password and noise level as locking time. Full password and noise details are available in special separate sections:

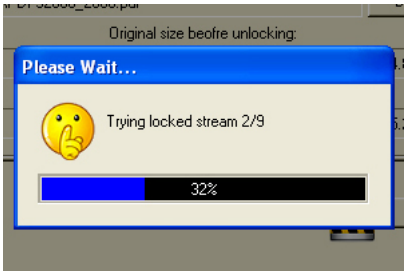
- EASY PASSWORDS SETUP
- OPTIONS: NOISE LEVEL

STEP 2:



(*Browse*)	Select a locked file
(*Original size before unlocking*)	Example: 89.954.880 bytes
(*Estimated size after unlocking*)	Example: 8.995.224 bytes
(*Unlock!*)	Start unlocking

Choose the locked data you want to unlock. Locked data will not be overwritten and unlocked secret data will be saved to a different folder.



Aspect number: (960 / Data) – 1
-1 because of χ^2-self-adjustment

Noise Level	Noise	Data	Aspects
300%	720	**240**	4 - 1
400%	768	**192**	5 - 1
500%	800	**160**	6 - 1
900%	864	**96**	10 - 1
1100%	880	**80**	12 - 1
1400%	896	**64**	15 - 1
1900%	912	**48**	20 - 1
2900%	928	**32**	30 - 1
5900%	944	**16**	60 - 1

Unlocking, even when passwords and locked data are ok, may take a long time due to the aspect number. The higher the noise level is, the more the aspects are. MultiObfuscator, by design, doesn't know which aspect was selected at locking time and has to slowly guess it by trial and error.

FEATURES: PROGRAM ARCHITECTURE

BACK

FILE LOCK – MEDIUM SETUP (4 PASSWORDS)

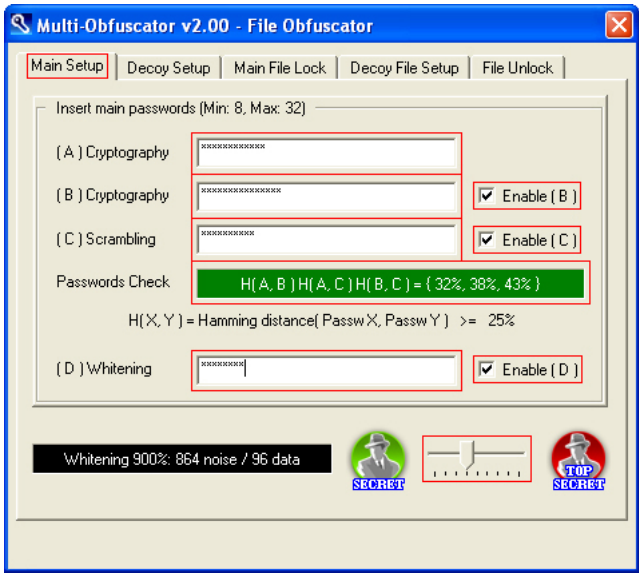
BEGIN:

(*File Lock/Unlock*)	Go to file (binary raw format) panel

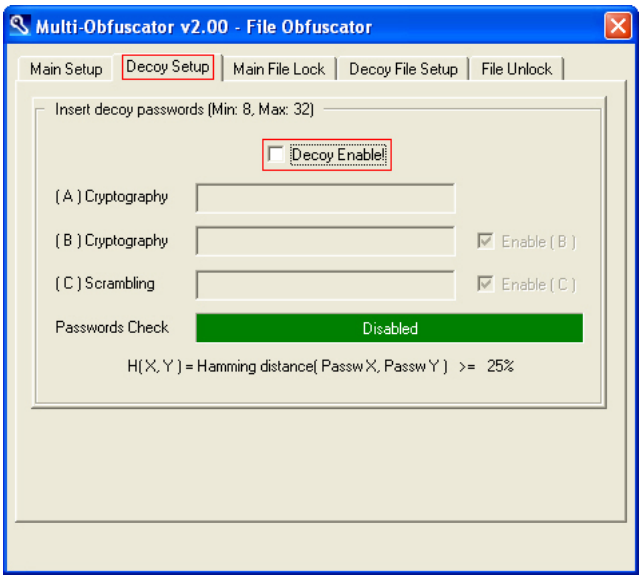
Select *File Lock/Unlock*.

STEP 1:

(I)



(II)



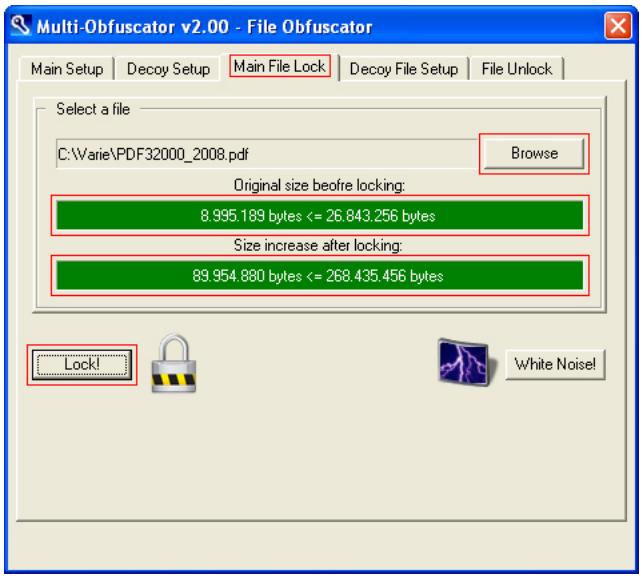
(I)	(*Cryptography A*)	First password
	(*Cryptography B*)	Second password (cryptography CSPRNG)
	(*Scrambling C*)	Third password (scrambling CSPRNG)
	(*Whitening D*)	Forth password (whitening CSPRNG)
	(*Enable B*)	Second password enable/disable
	(*Enable C*)	Third password enable/disable
	(*Enable D*)	Forth password enable/disable
(II)	(*Decoy Enable!*)	Decoy enable/disable

Insert a set of passwords and choose a noise level. Full password and noise details are available in special separate sections:

- MEDIUM PASSWORDS SETUP
- OPTIONS: NOISE LEVEL

Medium setup allows full usage of the multi-layered security architecture.
FEATURES: PROGRAM ARCHITECTURE

STEP 2:



(*Browse*)	Select a file
(*Original size before locking*)	Example: 8.995.189 bytes
(*Size increase after locking*)	Example: 89.954.880 bytes
(*Lock!*)	Start locking

Choose the secret data you want to lock (a single file or a zip/rar/… archive). Secret data will not be overwritten and locked data will be saved to a different folder. File/archive name will not be saved to the locked data, allowing renaming and unlocking secret data with a different name.

Example:

- MultiObfuscator: C:\…\dir1\xxx.pdf [9 Mb] → C:\...\dir2\xxx.pdf [90 Mb]
- Rename: C:\...\dir2\xxx.pdf → UsbKey:\...\yyy.pdf
- MultiObfuscator: UsbKey:\…\yyy.pdf [90 Mb] → D:\…\yyy.pdf [9 Mb]

There's a maximum locked size constraint of 256 Mb and, depending on the noise level, there's also a maximum plain size constraint. Little files (up to 4 Mb) will let you free to choose any noise level. Medium and large files (up to 64 Mb) will force you to choose a lower compatible (by size) noise level.

Example:

- Noise level: 900%
- Original size before locking: 8.995.189 bytes ≤ 25 Mb
- Size after locking: ((8.995.189 + 256) / 96) * 960 = 89.954.880 bytes ≤ 256 Mb

Noise Level	Noise	Data	Min. Plain → Locked Size	Max. Plain → Locked Size
900%	864	**96**	1 B → 2880 B	**25 Mb → 256 Mb**

OPTIONS: NOISE LEVEL

BACK

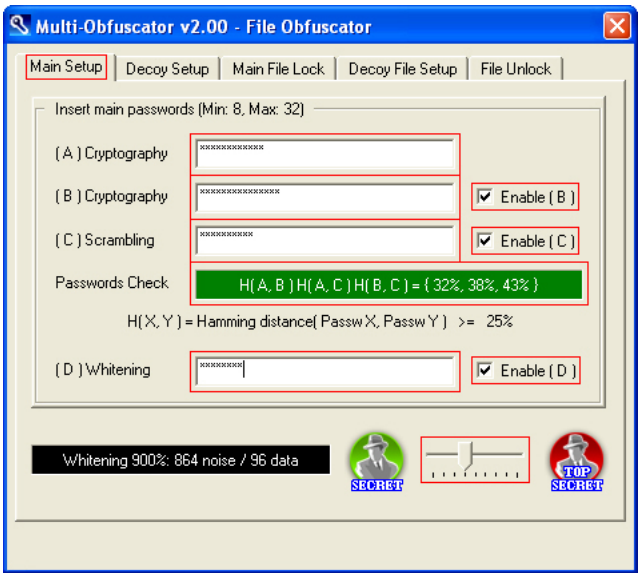
FILE UNLOCK – MEDIUM SETUP (4 PASSWORDS)

BEGIN:

(*File Lock/Unlock*)	Go to file (binary raw format) panel

Select *File Lock/Unlock*.

STEP 1:

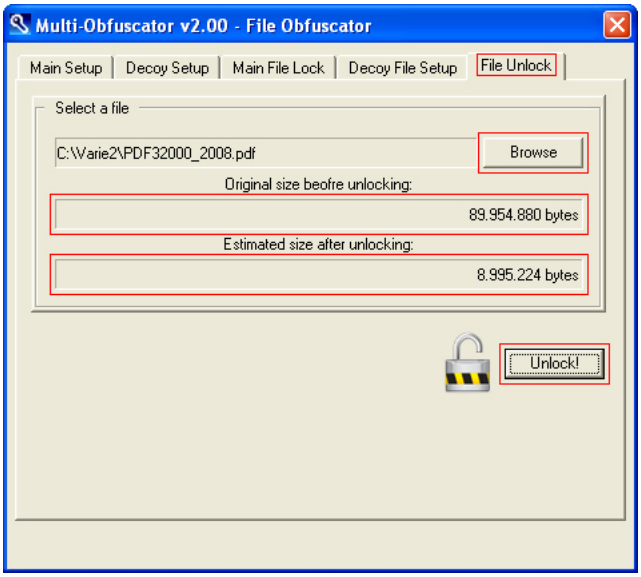


(*Cryptography A*)	First password
(*Cryptography B*)	Second password (cryptography CSPRNG)
(*Scrambling C*)	Third password (scrambling CSPRNG)
(*Whitening D*)	Forth password (whitening CSPRNG)
(*Enable B*)	Second password enable/disable
(*Enable C*)	Third password enable/disable
(*Enable D*)	Forth password enable/disable

Set same set of passwords and noise level as locking time. Full password and noise details are available in special separate sections:

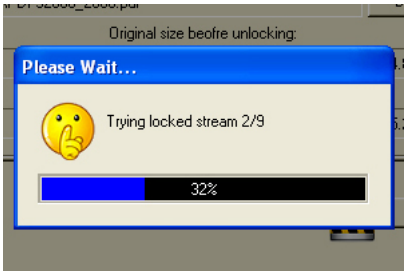
- MEDIUM PASSWORDS SETUP
- OPTIONS: NOISE LEVEL

STEP 2:



(*Browse*)	Select a locked file
(*Original size before unlocking*)	Example: 89.954.880 bytes
(*Estimated size after unlocking*)	Example: 8.995.224 bytes
(*Unlock!*)	Start unlocking

Choose the locked data you want to unlock. Locked data will not be overwritten and unlocked secret data will be saved to a different folder.



Aspect number: (960 / Data) – 1
-1 because of χ^2-self-adjustment

Noise Level	Noise	Data	Aspects
300%	720	**240**	4 - 1
400%	768	**192**	5 - 1
500%	800	**160**	6 - 1
900%	864	**96**	10 - 1
1100%	880	**80**	12 - 1
1400%	896	**64**	15 - 1
1900%	912	**48**	20 - 1
2900%	928	**32**	30 - 1
5900%	944	**16**	60 - 1

Unlocking, even when passwords and locked data are ok, may take a long time due to the aspect number. The higher the noise level is, the more the aspects are. MultiObfuscator, by design, doesn't know which aspect was selected at locking time and has to slowly guess it by trial and error.

FEATURES: PROGRAM ARCHITECTURE

BACK

EXPERT

FILE LOCK – ADVANCED SETUP (4 PASSWORDS+DECOY)

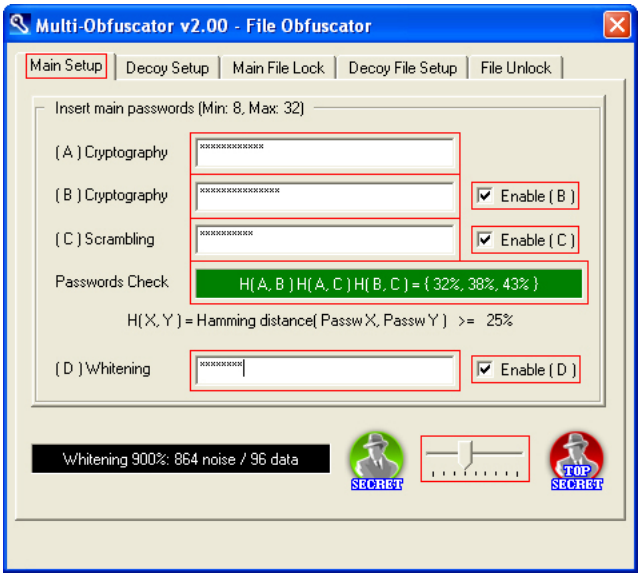
BEGIN:

(*File Lock/Unlock*)	Go to file (binary raw format) panel

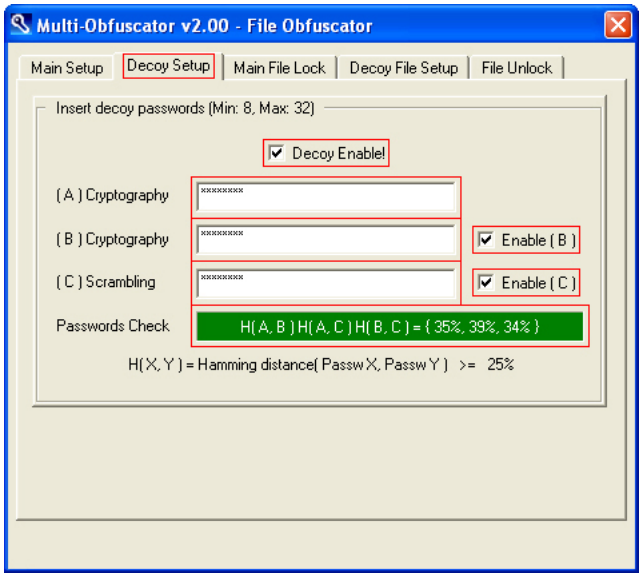
Select *File Lock/Unlock*.

STEP 1:

(I)



(II)



(I)	(*Cryptography A*)	First password
	(*Cryptography B*)	Second password (cryptography CSPRNG)
	(*Scrambling C*)	Third password (scrambling CSPRNG)
	(*Whitening D*)	Forth password (whitening CSPRNG)
	(*Enable B*)	Second password enable/disable
	(*Enable C*)	Third password enable/disable
	(*Enable D*)	Forth password enable/disable
(II)	(*Decoy Enable!*)	Decoy enable/disable
	(*Cryptography A*)	First decoy password
	(*Cryptography B*)	Second decoy password
	(*Scrambling C*)	Third decoy password
	(*Enable B*)	Second decoy password enable/disable
	(*Enable C*)	Third decoy password enable/disable

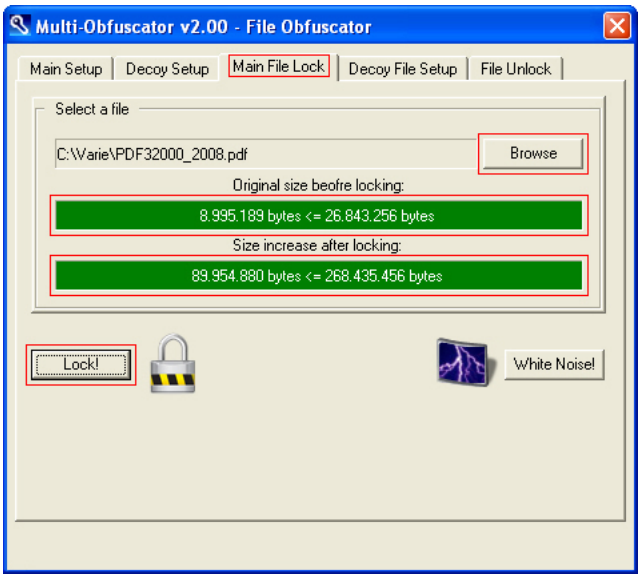
Insert a set of passwords, a set of decoy passwords and choose a noise level. Full password and noise details are available in special separate sections:

- ADVANCED PASSWORDS SETUP – LOCK
- OPTIONS: NOISE LEVEL

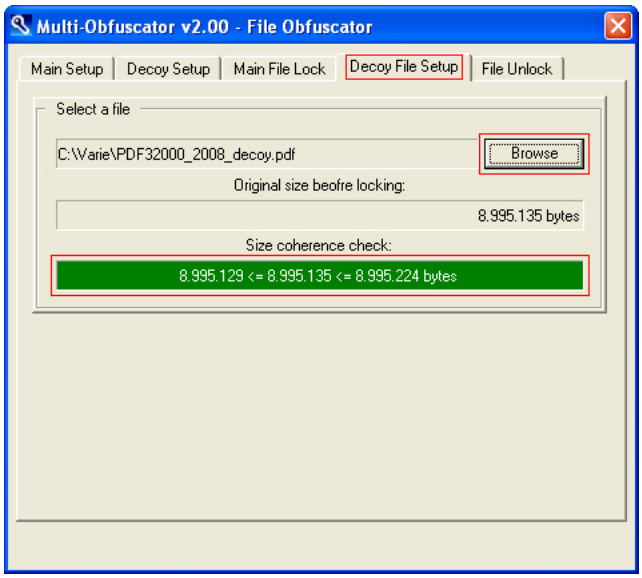
Advanced setup allows full usage of the multi-layered and multi-aspect security architecture.
FEATURES: PROGRAM ARCHITECTURE

STEP 2:

(I)



(II)



(I)	(*Browse*)	Select a file
	(*Original size before locking*)	Example: 8.995.189 bytes
	(*Size increase after locking*)	Example: 89.954.880 bytes
	(*Lock!*)	Start locking
(II)	(*Browse*)	Select a decoy file
	(*Size coherence check*)	Example: 8.995.135 bytes

Choose the secret data and a compatible (by size) decoy data you want to lock (a single file or a zip/rar/… archive).

Example:

- Noise level: 900%
- Original size before locking: 8.995.189 bytes ≤ 25 Mb
- Size after locking: ((8.995.189 + 256) / 96) * 960 = 89.954.880 bytes ≤ 256 Mb
- Decoy size: (((8.995.129 ≤ x ≤ 8.995.224) + 256) / 96) * 960 = 89.954.880 bytes ≤ 256 Mb

Noise Level	Noise	Data	Min. Plain → Locked Size	Max. Plain → Locked Size
900%	864	**96**	1 B → 2880 B	**25 Mb → 256 Mb**

Be aware that:
- the higher the noise level is, the less the data bytes per block are
- the less the data bytes per block are, the narrower the decoy size range is

Minimum (300%) → Data = 240 → inf ≤ x ≤ sup → sup - inf + 1 = 240 bytes
Maximum (5900%) → Data = 16 → inf ≤ x ≤ sup → sup - inf + 1 = 16 bytes

Be sure to read also the intermediate section
FILE LOCK – MEDIUM SETUP (4 PASSWORDS)

BACK

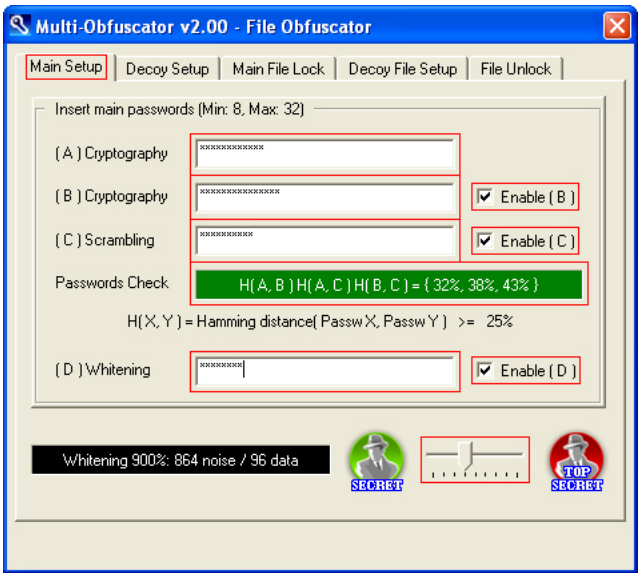
FILE UNLOCK – ADVANCED SETUP (4 PASSWORDS+DECOY)

BEGIN:

(*File Lock/Unlock*)	Go to file (binary raw format) panel

Select *File Lock/Unlock*.

STEP 1:



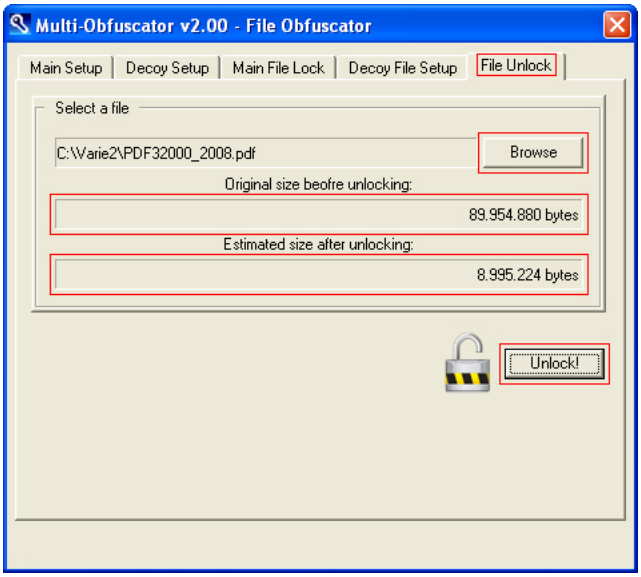
(*Cryptography A*)	First password
(*Cryptography B*)	Second password (cryptography CSPRNG)
(*Scrambling C*)	Third password (scrambling CSPRNG)
(*Whitening D*)	Forth password (whitening CSPRNG)
(*Enable B*)	Second password enable/disable
(*Enable C*)	Third password enable/disable
(*Enable D*)	Forth password enable/disable

Set same set of passwords (secret to get secret data, decoy to get decoy data) and noise level as locking time. Full password and noise details are available in special separate sections:

- ADVANCED PASSWORDS SETUP – UNLOCK
- OPTIONS: NOISE LEVEL

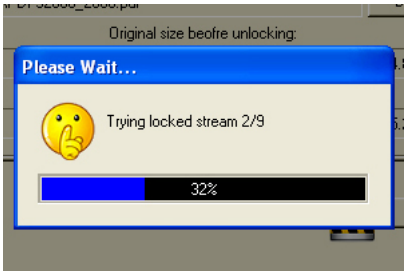
Detailed decoy details are available here:
WHAT IS DENIABLE CRYPTOGRAPHY?

STEP 2:



(*Browse*)	Select a locked file
(*Original size before unlocking*)	Example: 89.954.880 bytes
(*Estimated size after unlocking*)	Example: 8.995.224 bytes
(*Unlock!*)	Start unlocking

Choose the locked data you want to unlock. Locked data will not be overwritten and unlocked data (secret or decoy, depending on the set of passwords) will be saved to a different folder.



Aspect number: (960 / Data) – 1
-1 because of χ^2-self-adjustment

Noise Level	Noise	Data	Aspects
300%	720	**240**	4 - 1
400%	768	**192**	5 - 1
500%	800	**160**	6 - 1
900%	864	**96**	10 - 1
1100%	880	**80**	12 - 1
1400%	896	**64**	15 - 1
1900%	912	**48**	20 - 1
2900%	928	**32**	30 - 1
5900%	944	**16**	60 - 1

Unlocking, even when passwords and locked data are ok, may take a long time due to the aspect number. The higher the noise level is, the more the aspects are. MultiObfuscator, by design, doesn't know which aspect was selected at locking time and has to slowly guess it by trial and error.

FEATURES: PROGRAM ARCHITECTURE

BACK

EXPERT   

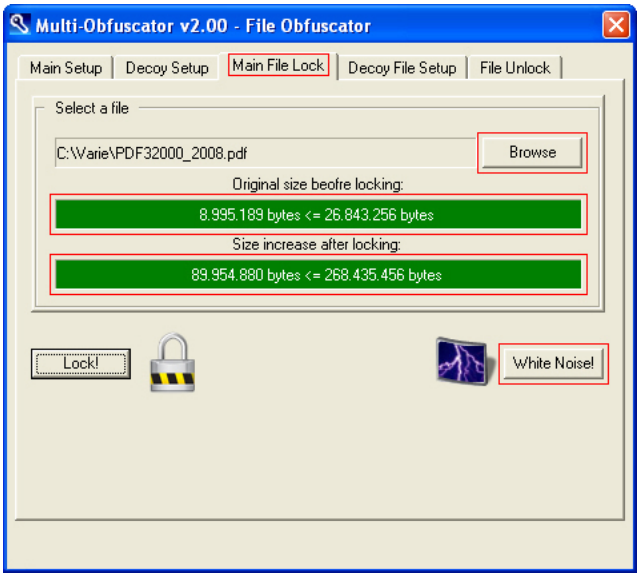
WHITE NOISE AS A DECOY (FILE)

BEGIN:

(*File Lock/Unlock*)	Go to file (binary raw format) panel

Select *File Lock/Unlock*.

STEP 1:



(*Browse*)	Select a file
(*Original size before locking*)	Example: 8.995.189 bytes
(*Size increase after locking*)	Example: 89.954.880 bytes
(*White Noise!*)	Start randomizing

Locked files are statistically undistinguishable from void randomized files. Advanced users will be able to add void/fake containers to the sentive ones, in order to waste attackers' time. This task will save white noise only to a fake container compatible (by size) with the selected file.

FEATURES: PROGRAM ARCHITECTURE

Example:

- Noise level: 900%
- Size after locking: ((8.995.189 + 256) / 96) * 960 = **89.954.880** bytes ≤ 256 Mb
- White noise size: **89.954.880** bytes

Noise Level	Noise	Data	Min. Plain → Locked Size	Max. Plain → Locked Size
900%	864	**96**	1 B → 2880 B	25 Mb → 256 Mb

OPTIONS: NOISE LEVEL

BACK

EASY	

TEXT LOCK – BASE SETUP (1 PASSWORD)

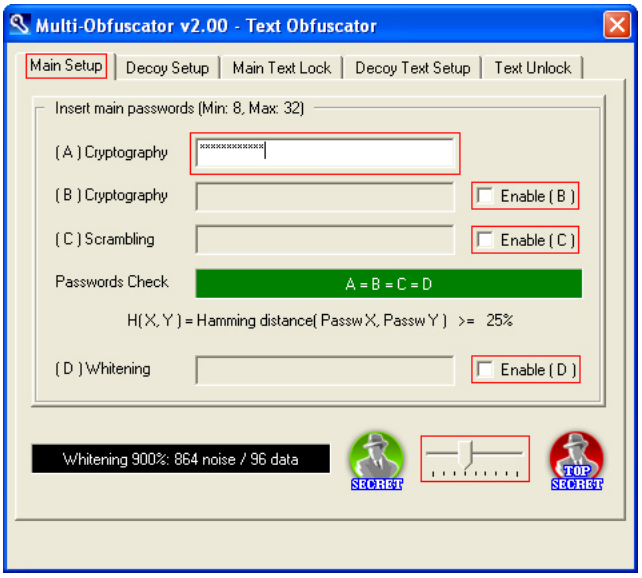
BEGIN:

(*Text Lock/Unlock*)	Go to text (email format) panel

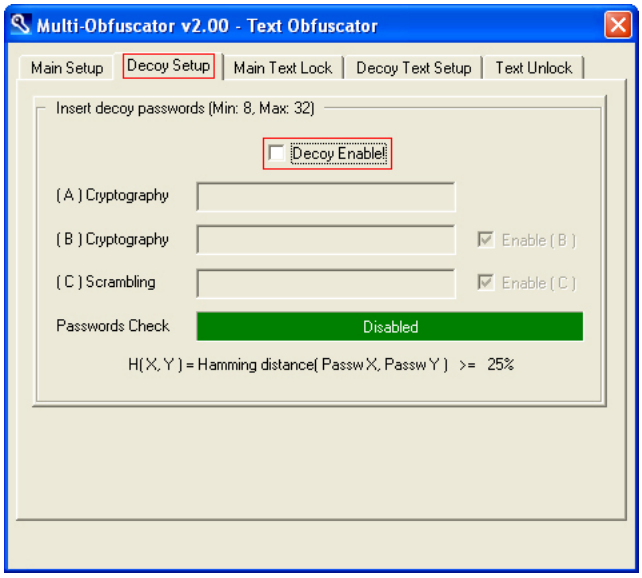
Select *Text Lock/Unlock*.

STEP 1:

(I)



(II)



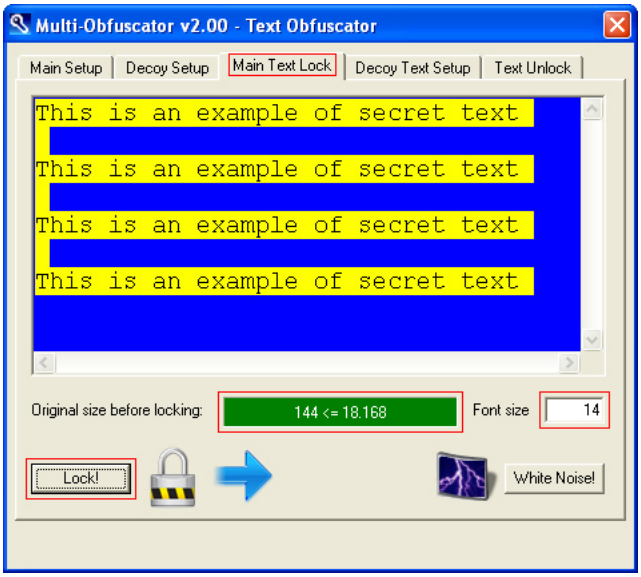
(I)	(*Cryptography A*)	First password
	(*Enable B*)	Second password enable/disable
	(*Enable C*)	Third password enable/disable
	(*Enable D*)	Forth password enable/disable
(II)	(*Decoy Enable!*)	Decoy enable/disable

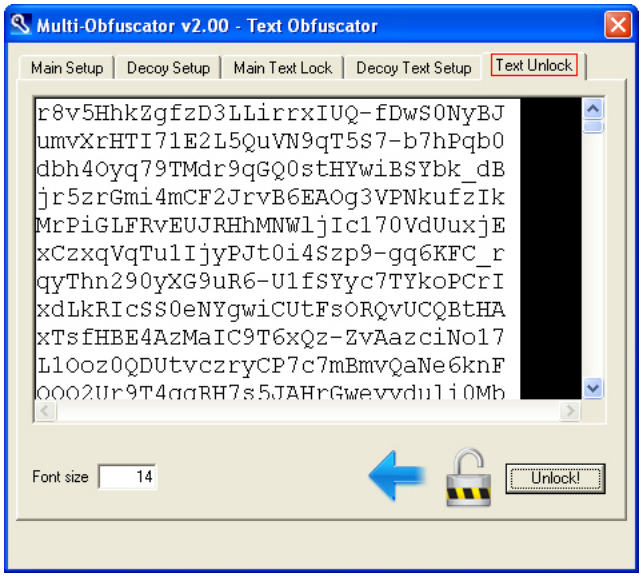
Insert a password and choose a noise level. Full password and noise details are available in special separate sections:

- EASY PASSWORDS SETUP
- OPTIONS: NOISE LEVEL

Base setup, even though looking like a traditional security software, relies on the same multi-layered security architecture as advanced setup.
FEATURES: PROGRAM ARCHITECTURE

STEP 2:

(I) 

(II) 

(I)	< TextEdit – blue window >	Enter/paste a text
	(*Original size before locking*)	Example: 144 bytes
	(*Font size*)	Text font size
	(*Lock!*)	Start locking

Choose the secret text you want to lock. Secret text will not be overwritten and locked text will be saved to the *Text Unlock* window, ready to be cut and pasted.

There's a maximum locked size constraint of 256 Kb that, depending on the noise level, will also add a maximum plain size constraint. Little files (up to 3 Kb) will let you free to choose any noise level. Medium and large files (up to 46 Kb) will force you to choose a lower compatible (by size) noise level.

Example:

- Noise level: 900%
- Original size before locking: 144 bytes ≤ 18 Kb
- Size after locking: ((144 + 256) / 96) * 1280 = 6.400 bytes ≤ 256 Kb

Noise Level	Noise	Data	Min. Plain → Locked Size	Max. Plain → Locked Size
900%	864	**96**	1 B → 3840 B	**18 Kb → 256 Kb**

OPTIONS: NOISE LEVEL

BACK

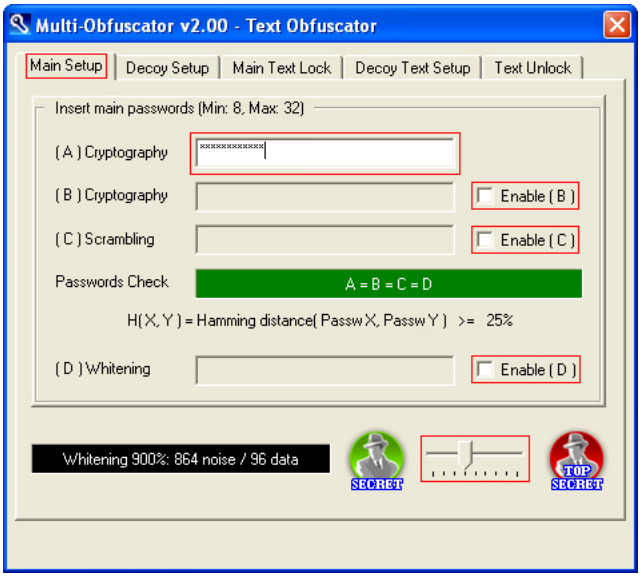
TEXT UNLOCK – BASE SETUP (1 PASSWORD)

BEGIN:

(*Text Lock/Unlock*)	Go to text (email format) panel

Select *Text Lock/Unlock*.

STEP 1:



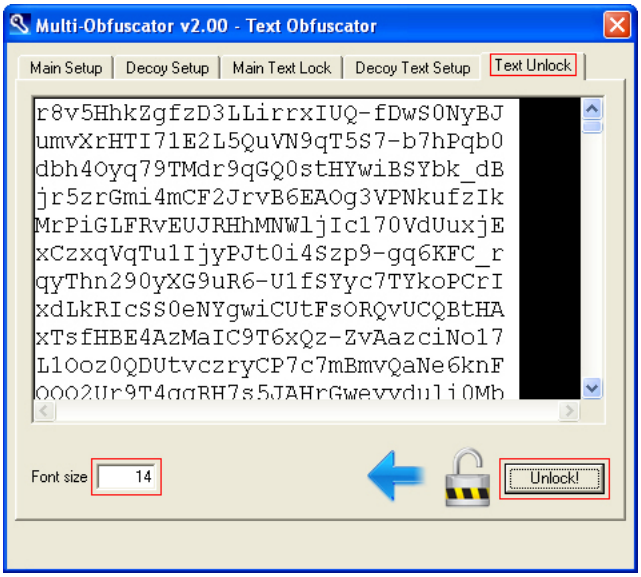
(*Cryptography A*)	First password
(*Enable B*)	Second password enable/disable
(*Enable C*)	Third password enable/disable
(*Enable D*)	Forth password enable/disable

Set same password and noise level as locking time. Full password and noise details are available in special separate sections:

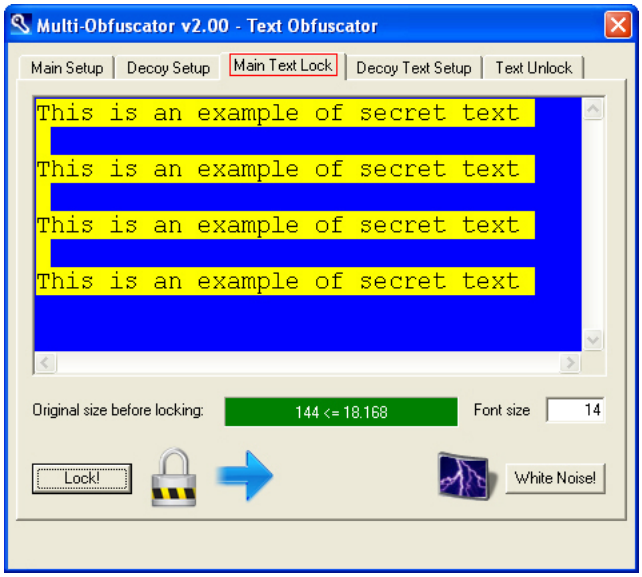
- EASY PASSWORDS SETUP
- OPTIONS: NOISE LEVEL

STEP 2:

(I)

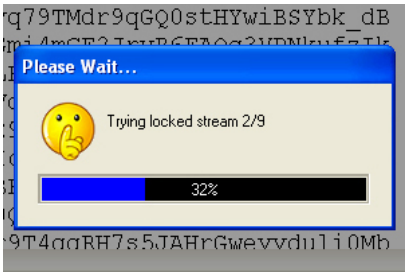


(II)



(I)	< TextEdit – black window >	Enter/paste a locked text
	(*Font size*)	Text font size
	(*Unlock!*)	Start unlocking

Choose the locked text you want to unlock. Locked text will not be overwritten and unlocked secret text will be saved to the *Main Text Lock* window, ready to be cut and pasted.



Aspect number: (960 / Data) – 1
-1 because of χ^2-self-adjustment

Noise Level	Noise	Data	Aspects
300%	720	**240**	4 - 1
400%	768	**192**	5 - 1
500%	800	**160**	6 - 1
900%	864	**96**	10 - 1
1100%	880	**80**	12 - 1
1400%	896	**64**	15 - 1
1900%	912	**48**	20 - 1
2900%	928	**32**	30 - 1
5900%	944	**16**	60 - 1

Unlocking, even when passwords and locked text are ok, may take a long time due to the aspect number. The higher the noise level is, the more the aspects are. MultiObfuscator, by design, doesn't know which aspect was selected at locking time and has to slowly guess it by trial and error.

FEATURES: PROGRAM ARCHITECTURE

BACK

TEXT LOCK – MEDIUM SETUP (4 PASSWORDS)

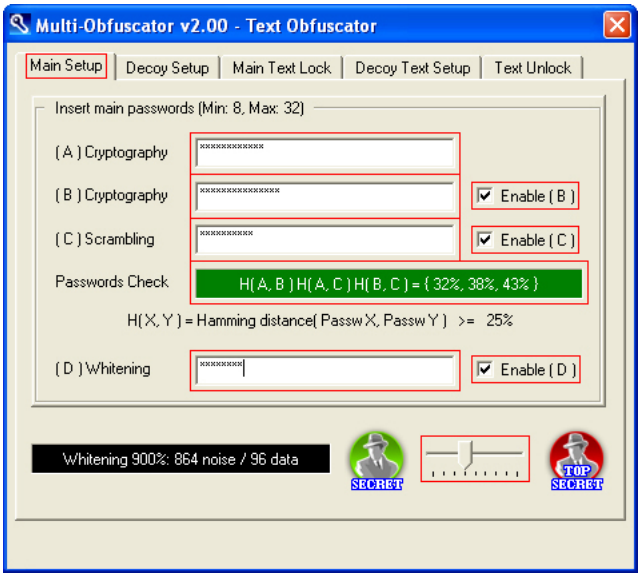
BEGIN:

(*Text Lock/Unlock*)	Go to text (email format) panel

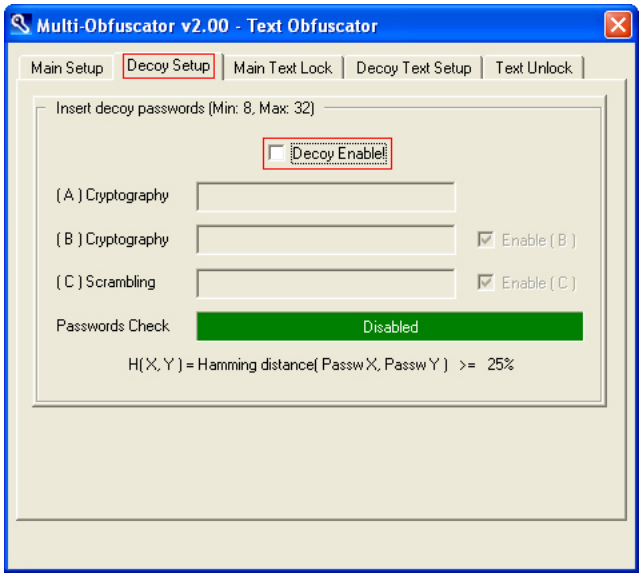
Select *Text Lock/Unlock*.

STEP 1:

(I)



(II)



(I)	(*Cryptography A*)	First password
	(*Cryptography B*)	Second password (cryptography CSPRNG)
	(*Scrambling C*)	Third password (scrambling CSPRNG)
	(*Whitening D*)	Forth password (whitening CSPRNG)
	(*Enable B*)	Second password enable/disable
	(*Enable C*)	Third password enable/disable
	(*Enable D*)	Forth password enable/disable
(II)	(*Decoy Enable!*)	Decoy enable/disable

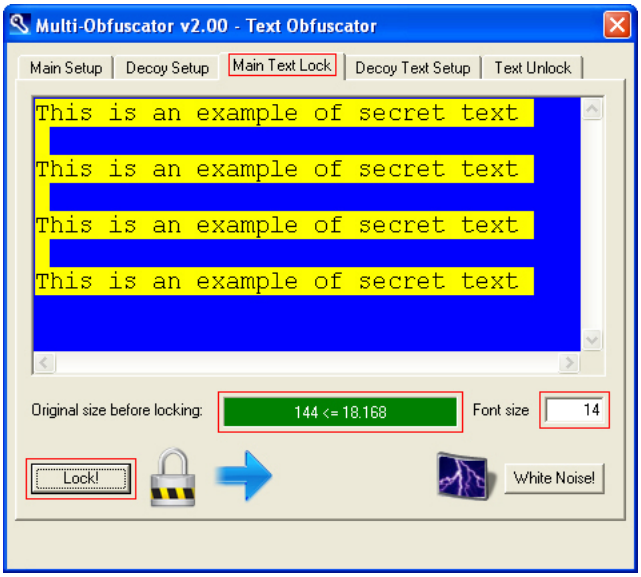
Insert a set of passwords and choose a noise level. Full password and noise details are available in special separate sections:

- MEDIUM PASSWORDS SETUP
- OPTIONS: NOISE LEVEL

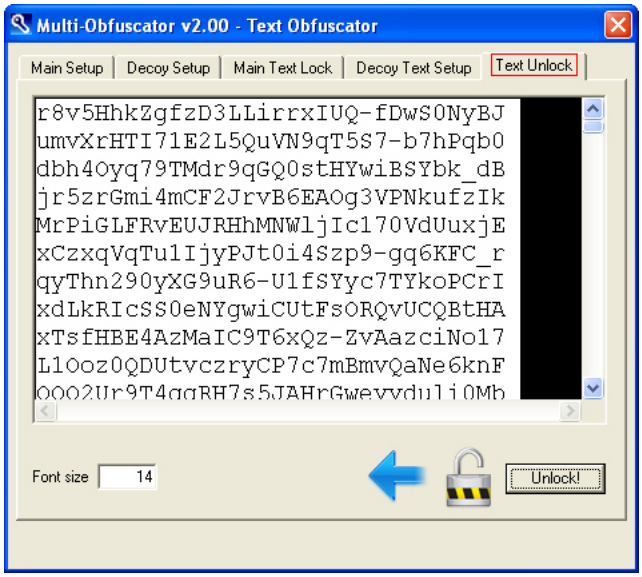
Medium setup allows full usage of the multi-layered security architecture.
FEATURES: PROGRAM ARCHITECTURE

STEP 2:

(I)



(II)



(I)	< TextEdit – blue window >	Enter/paste a text
	(*Original size before locking*)	Example: 144 bytes
	(*Font size*)	Text font size
	(*Lock!*)	Start locking

Choose the secret text you want to lock. Secret text will not be overwritten and locked text will be saved to the *Text Unlock* window, ready to be cut and pasted.

There's a maximum locked size constraint of 256 Kb that, depending on the noise level, will also add a maximum plain size constraint. Little files (up to 3 Kb) will let you free to choose any noise level. Medium and large files (up to 46 Kb) will force you to choose a lower compatible (by size) noise level.

Example:

- Noise level: 900%
- Original size before locking: 144 bytes ≤ 18 Kb
- Size after locking: ((144 + 256) / 96) * 1280 = 6.400 bytes ≤ 256 Kb

Noise Level	Noise	Data	Min. Plain → Locked Size	Max. Plain → Locked Size
900%	864	**96**	1 B → 3840 B	**18 Kb → 256 Kb**

OPTIONS: NOISE LEVEL

BACK

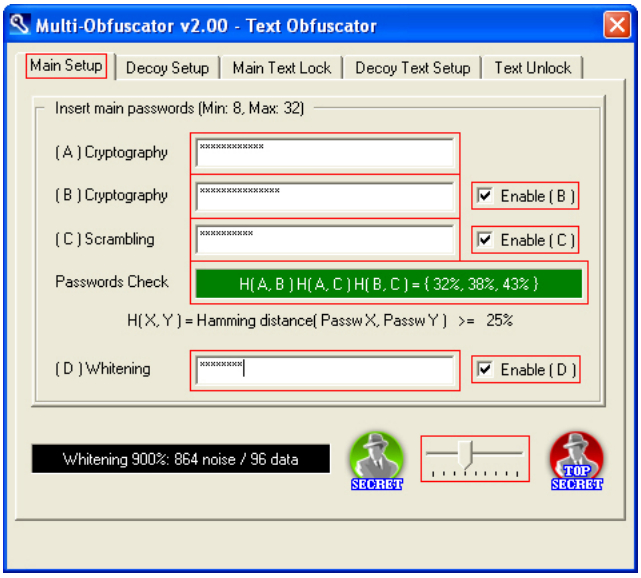
TEXT UNLOCK – MEDIUM SETUP (4 PASSWORDS)

BEGIN:

(*Text Lock/Unlock*)	Go to text (email format) panel

Select *Text Lock/Unlock*.

STEP 1:



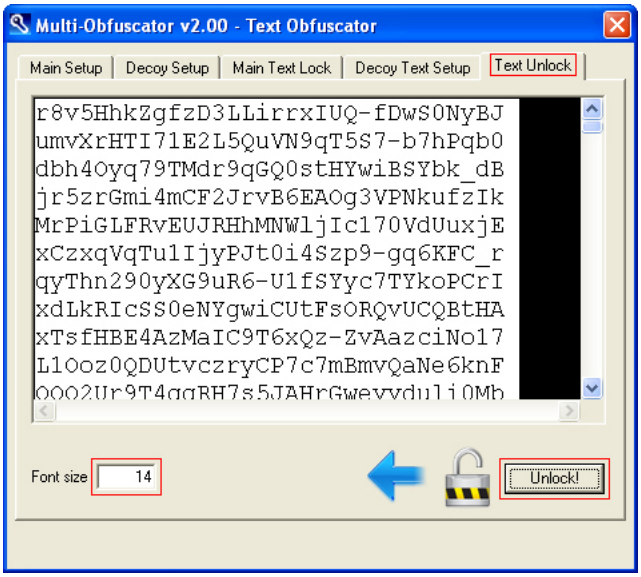
(*Cryptography A*)	First password
(*Cryptography B*)	Second password (cryptography CSPRNG)
(*Scrambling C*)	Third password (scrambling CSPRNG)
(*Whitening D*)	Forth password (whitening CSPRNG)
(*Enable B*)	Second password enable/disable
(*Enable C*)	Third password enable/disable
(*Enable D*)	Forth password enable/disable

Set same set of passwords and noise level as locking time. Full password and noise details are available in special separate sections:

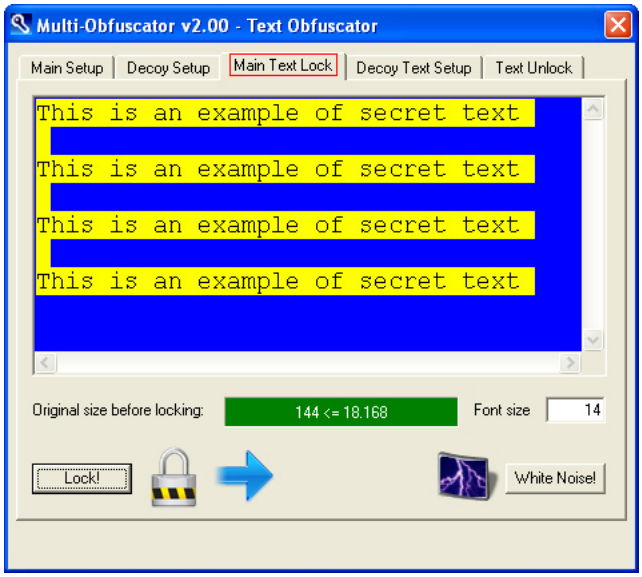
- MEDIUM PASSWORDS SETUP
- OPTIONS: NOISE LEVEL

STEP 2:

(I)

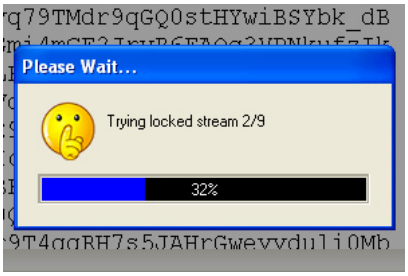


(II)



(I)	< TextEdit – black window >	Enter/paste a locked text
	(*Font size*)	Text font size
	(*Unlock!*)	Start unlocking

Choose the locked text you want to unlock. Locked text will not be overwritten and unlocked secret text will be saved to the *Main Text Lock* window, ready to be cut and pasted.



Aspect number: (960 / Data) – 1
-1 because of χ^2-self-adjustment

Noise Level	Noise	Data	Aspects
300%	720	**240**	4 - 1
400%	768	**192**	5 - 1
500%	800	**160**	6 - 1
900%	864	**96**	10 - 1
1100%	880	**80**	12 - 1
1400%	896	**64**	15 - 1
1900%	912	**48**	20 - 1
2900%	928	**32**	30 - 1
5900%	944	**16**	60 - 1

Unlocking, even when passwords and locked text are ok, may take a long time due to the aspect number. The higher the noise level is, the more the aspects are. MultiObfuscator, by design, doesn't know which aspect was selected at locking time and has to slowly guess it by trial and error.

FEATURES: PROGRAM ARCHITECTURE

BACK

EXPERT

TEXT LOCK – ADVANCED SETUP (4 PASSWORDS+DECOY)

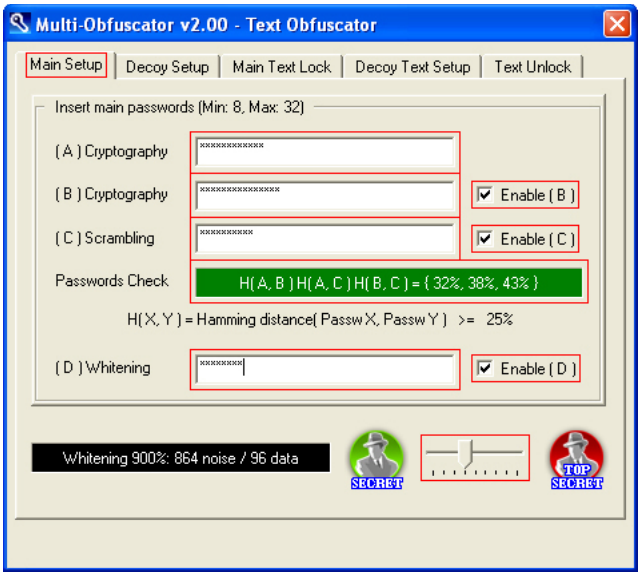
BEGIN:

(*Text Lock/Unlock*)	Go to text (email format) panel

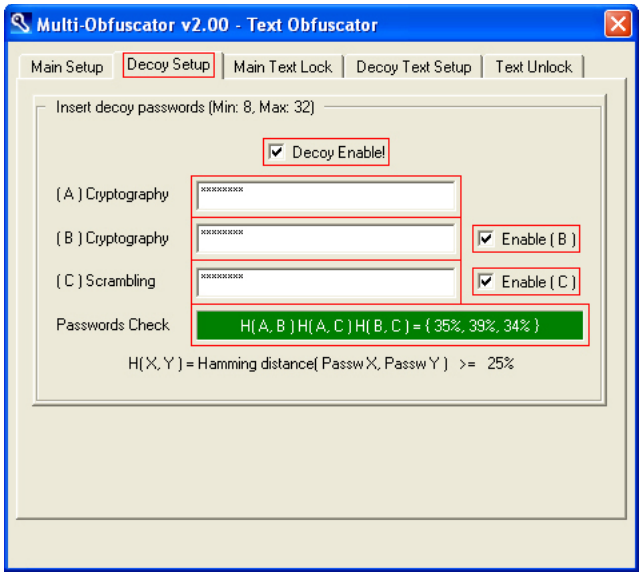
Select *Text Lock/Unlock*.

STEP 1:

(I)



(II)



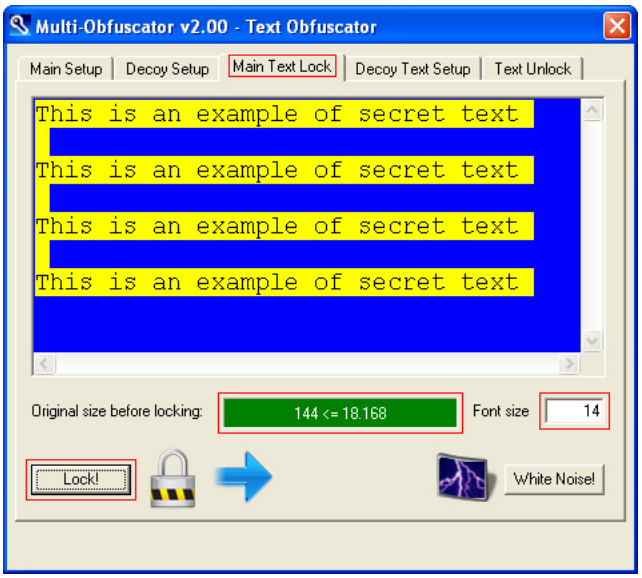
(I)	(*Cryptography A*)	First password
	(*Cryptography B*)	Second password (cryptography CSPRNG)
	(*Scrambling C*)	Third password (scrambling CSPRNG)
	(*Whitening D*)	Forth password (whitening CSPRNG)
	(*Enable B*)	Second password enable/disable
	(*Enable C*)	Third password enable/disable
	(*Enable D*)	Forth password enable/disable
(II)	(*Decoy Enable!*)	Decoy enable/disable
	(*Cryptography A*)	First decoy password
	(*Cryptography B*)	Second decoy password
	(*Scrambling C*)	Third decoy password
	(*Enable B*)	Second decoy password enable/disable
	(*Enable C*)	Third decoy password enable/disable

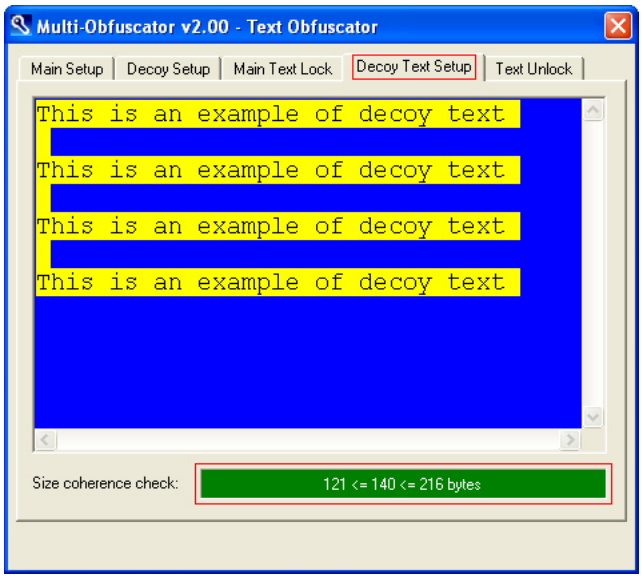
Insert a set of passwords, a set of decoy passwords and choose a noise level. Full password and noise details are available in special separate sections:

- ADVANCED PASSWORDS SETUP – LOCK
- OPTIONS: NOISE LEVEL

Advanced setup allows full usage of the multi-layered and multi-aspect security architecture.
FEATURES: PROGRAM ARCHITECTURE

STEP 2:

(I) 

(II) 

(I)	< TextEdit – blue window >	Enter/paste a text
	(*Original size before locking*)	Example: 144 bytes
	(*Font size*)	Text font size
	(*Lock!*)	Start locking
(II)	< TextEdit – blue window >	Enter/paste a decoy text
	(*Size coherence check*)	Example: 140 bytes

Choose the secret text and a compatible (by size) decoy text you want to lock.

Example:

- Noise level: 900%
- Original size before locking: 144 bytes ≤ 18 Kb
- Size after locking: ((144 + 256) / 96) * 1280 = 6.400 bytes ≤ 256 Kb
- Decoy size: (((121 ≤ x ≤ 216) + 256) / 96) * 1280 = 6.400 bytes ≤ 256 Kb

Noise Level	Noise	Data	Min. Plain → Locked Size	Max. Plain → Locked Size
900%	864	**96**	1 B → 3840 B	**18 Kb → 256 Kb**

Be aware that:
- the higher the noise level is, the less the data bytes per block are
- the less the data bytes per block are, the narrower the decoy size range is

Minimum (300%) → Data = 240 → inf ≤ x ≤ sup → sup - inf = 240 bytes
Maximum (5900%) → Data = 16 → inf ≤ x ≤ sup → sup - inf = 16 bytes

Be sure to read also the intermediate section
TEXT LOCK – MEDIUM SETUP (4 PASSWORDS)

BACK

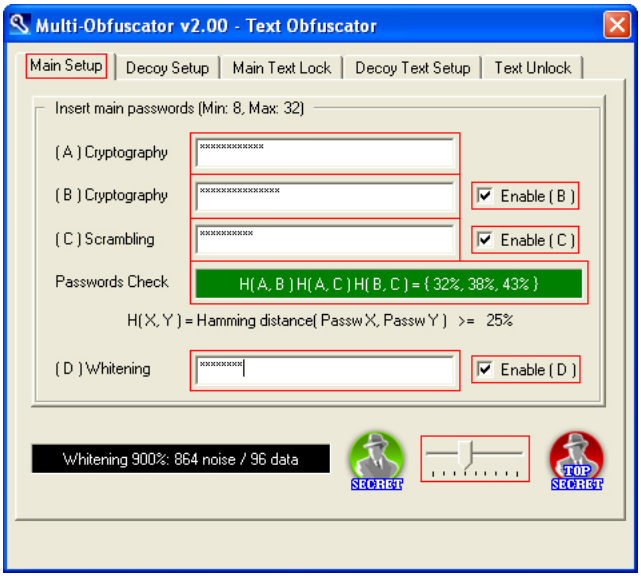
TEXT UNLOCK – ADVANCED SETUP (4 PASSWORDS+DECOY)

BEGIN:

(*Text Lock/Unlock*)	Go to text (email format) panel

Select *Text Lock/Unlock*.

STEP 1:



(*Cryptography A*)	First password
(*Cryptography B*)	Second password (cryptography CSPRNG)
(*Scrambling C*)	Third password (scrambling CSPRNG)
(*Whitening D*)	Forth password (whitening CSPRNG)
(*Enable B*)	Second password enable/disable
(*Enable C*)	Third password enable/disable
(*Enable D*)	Forth password enable/disable

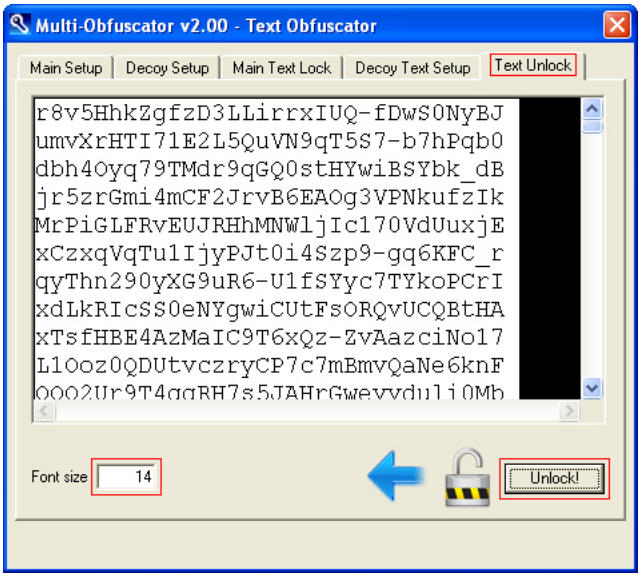
Set same set of passwords (secret to get secret data, decoy to get decoy data) and noise level as locking time. Full password and noise details are available in special separate sections:

- ADVANCED PASSWORDS SETUP – UNLOCK
- OPTIONS: NOISE LEVEL

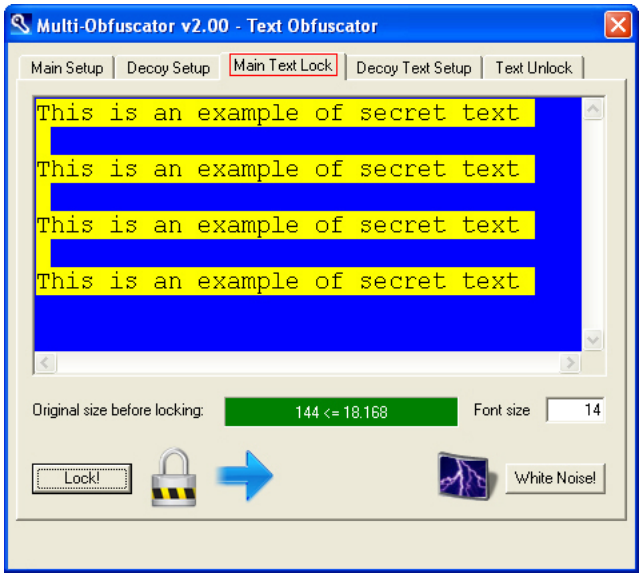
Detailed decoy details are available here:
WHAT IS DENIABLE CRYPTOGRAPHY?

STEP 2:

(I)

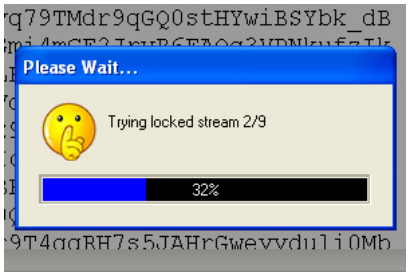


(II)



(I)	< TextEdit – black window >	Enter/paste a locked text
	(*Font size*)	Text font size
	(*Unlock!*)	Start unlocking

Choose the locked text you want to unlock. Locked text will not be overwritten and unlocked text (secret or decoy, depending on the set of passwords) will be saved to the *Main Text Lock* window, ready to be cut and pasted.



Aspect number: (960 / Data) – 1
-1 because of χ^2-self-adjustment

Noise Level	Noise	Data	Aspects
300%	720	**240**	4 - 1
400%	768	**192**	5 - 1
500%	800	**160**	6 - 1
900%	864	**96**	10 - 1
1100%	880	**80**	12 - 1
1400%	896	**64**	15 - 1
1900%	912	**48**	20 - 1
2900%	928	**32**	30 - 1
5900%	944	**16**	60 - 1

Unlocking, even when passwords and locked text are ok, may take a long time due to the aspect number. The higher the noise level is, the more the aspects are. MultiObfuscator, by design, doesn't know which aspect was selected at locking time and has to slowly guess it by trial and error.

FEATURES: PROGRAM ARCHITECTURE

BACK

EXPERT		

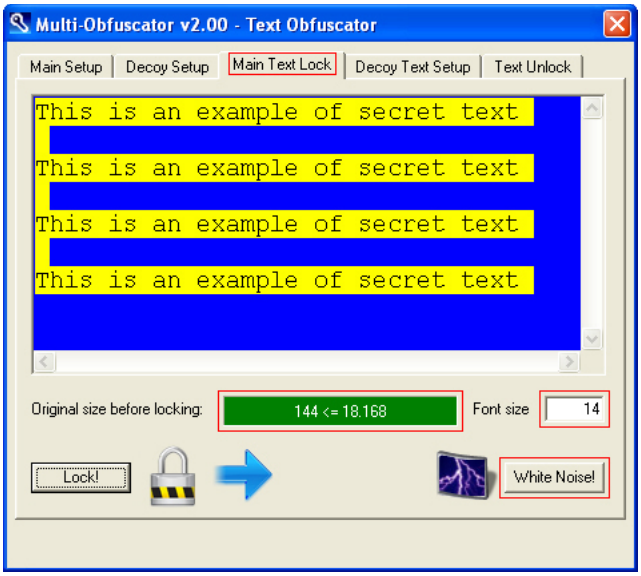
WHITE NOISE AS A DECOY (TEXT)

BEGIN:

(*Text Lock/Unlock*)	Go to text (email format) panel

Select *Text Lock/Unlock*.

STEP 1:



< TextEdit – blue window >	Enter/paste a text
(*Original size before locking*)	Example: 144 bytes
(*Font size*)	Text font size
(*White Noise!*)	Start randomizing

Locked text is statistically undistinguishable from void randomized text. Advanced users will be able to add void/fake texts to the sentive ones, in order to waste attackers' time. This task will save white noise only to a fake container compatible (by size) with the selected text.

FEATURES: PROGRAM ARCHITECTURE

Example:

- Noise level: 900%
- Size after locking: ((144 + 256) / 96) * 1280 = **6.400** bytes ≤ 256 Kb
- White noise size: **6.400** bytes

Noise Level	Noise	Data	Min. Plain → Locked Size	Max. Plain → Locked Size
900%	864	**96**	1 B → 3840 B	18 Kb → 256 Kb

OPTIONS: NOISE LEVEL

BACK