FooCrypt, A Tale of Cynical Cyclical Encryption.

Cryptography & Steganography

White Paper

- **FooCrypt**
  - **Versions**
    - X.Y.Z
      - [Current FooCrypt Release]
      - Where :
        - X = Major Release Number [UPGRADE]
        - Y = Minor Release Number [UPDATE]
        - Z = Patch Release Number [FIX]
    - XX.YY.ZZ
      - [Scheduled For 3rd Quarter 2024 Release]
      - Where :
        - XX = Major Release Number [UPGRADE]
        - YY = Minor Release Number [UPDATE]
        - ZZ = Patch Release Number [FIX]
  - **Release**
    - 11.0.0.Core
  - **Containing** :
    - FooCheck FooCrypt FooCrypt-Desktop FooCrypt-GUI FooCrypt-GUI-Data-1
    - FooCrypt-GUI-Data-2 FooSteg CLI_Test Decrypt_FooKey mOpenSSL
    - mFooKey Matrix_Test
    - 11.0.0.OpenSSL
      - precompiled mOpenSSL OpenSSL versions via : foocrypt-11.0.0-openssl-linux_x86_64.deb with OpenSSL :

- /opt/FooCrypt-OpenSSL/Linux/bin_64/openssl-1.1.1w/bin/openssl : OpenSSL 1.1.1w 11 Sep 2023
- /opt/FooCrypt-OpenSSL/Linux/bin_64/openssl-3.0.12/bin/openssl : OpenSSL 3.0.12 24 Oct 2023 (Library: OpenSSL 3.0.12 24 Oct 2023)
- /opt/FooCrypt-OpenSSL/Linux/bin_64/openssl-3.1.4/bin/openssl : OpenSSL 3.1.4 24 Oct 2023 (Library: OpenSSL 3.1.4 24 Oct 2023)
- /opt/FooCrypt-OpenSSL/Linux/bin_64/openssl-3.2.0/bin/openssl : OpenSSL 3.2.0 23 Nov 2023 (Library: OpenSSL 3.2.0 23 Nov 2023)

  - **Operating Systems**
    - Darwin (macOS)
    - Linux (DEBIAN & RHEL Packaging Formats)
    - Live Linux (Any OS running a Hypervisor) Based on uBuntu 22.04.3 LTS
    - SunOS (Solaris x86 & OpenIndiana)
    - Windows 10+ running the W.S.L.2.+

Documentation Version

- [20240126115959]

CopyRights

Audience

- The 'FooCrypt, A Tale of Cynical Cyclical Encryption, Cryptography & Steganography White Paper' aims to deliver an understanding of the technical details, specifications, process's, methods, etc that are utilised within the 'FooCrypt, A Tale of Cynical Cyclical Encryption' toolkit to the reader.

- The intended audience level, is focused towards a 'lay person', and also provides details which are focused towards a 'technical person'.

- This White Paper includes proprietary terms and concepts, unique to the 'FooCrypt, A Tale of Cynical Cyclical Encryption' toolkit and QRCrypto's eAES® Quantum Resistant Cipher Engine, which may require further understanding from the reader.

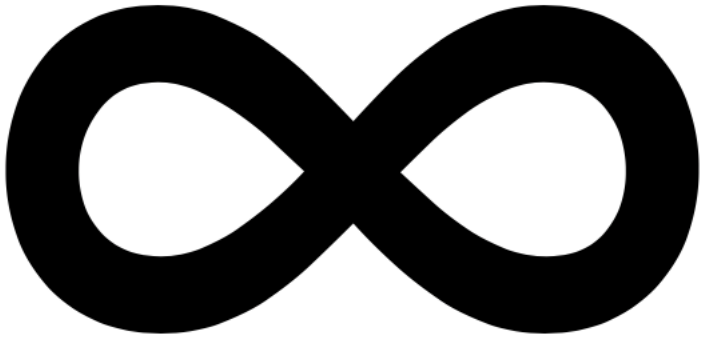
- An evolving **Acronyms and Abbreviations** section is available to assist with the readers understanding of the proprietary terms and concepts.

Contents

Introduction

‘FooCrypt makes a mountain out of a mole hill, and FooSteg hides the mountain in plain sight’

Protecting Data via An Effectively Infinite Brute Force & Obfuscation Solution(1)



(1) Lay persons terms representation of calculations covering combinations of a Data Image to recover the Secret Data Blob. Infinity symbol utilised to represent compute power verses time, utilising known side channel attacks methods, brute force attacks methods, et al.

Licensing Software

Software License – License for `FooCrypt, A Tale Of Cynical Cyclical Encryption.`

License Summary

Cannot modify source-code for any purpose (cannot create derivative works)
Support provided
License does not expire.
Commercial use allowed

`FooCrypt, A Tale Of Cynical Cyclical Encryption.` – Terms and conditions

1. Preamble: This Agreement, signed on Feb 1, 2018 (hereinafter: Effective Date) governs the relationship between `PURCHASER`, (hereinafter: Licensee) and Cryptopocalypse, a duly registered company in whose principal place of business is P.O. Box 333, Pyalong. VIC 3521. Australia. (hereinafter: Licensor). This Agreement sets the terms, rights, restrictions and obligations on using `FooCrypt, A Tale Of Cynical Cyclical Encryption.` (hereinafter: The Software) created and owned by Licensor, as detailed herein

2. License Grant: Licensor hereby grants Licensee a Personal, Non-assignable & non-transferable, Perpetual, Commercial, Royalty free, Without the rights to create derivative works, Non-exclusive license, all with accordance with the terms set forth and other legal restrictions set forth in 3rd party software used while running Software.

2.1. Limited: Licensee may use Software for the purpose of:

2.1.1. Running Software on Licensee's Website[s] and Server[s] and Work Station[s];

2.1.2. Allowing 3rd Parties to run Software on Licensee's Website[s] and Server[s] and Work Station[s];

2.1.3. Publishing Software's output to Licensee and 3rd Parties;

2.1.4. Distribute verbatim copies of Software's output (including compiled binaries);

2.2. This license is granted perpetually, as long as you do not materially breach it.

2.3. Non Assignable & Non-Transferable: Licensee may not assign or transfer his rights and duties under this license.

2.4. Commercial, Royalty Free: Licensee may use Software for any purpose, including paid-services, without any royalties

2.6. With support & maintenance: Licensor shall provide Licensee support and maintenance as follows in 5.0 through 6.2

3. Term & Termination: The Term of this license shall be until terminated. Licensor may terminate this Agreement, including Licensee's license in the case where Licensee :

3.1. became insolvent or otherwise entered into any liquidation process; or

3.2. exported The Software to any jurisdiction where licensor may not enforce his rights under this agreements in; or

3.3. Licensee was in breach of any of this license's terms and conditions and such breach was not cured, immediately upon notification; or

3.4. Licensee in breach of any of the terms of clause 2 to this license; or

3.5. Licensee otherwise entered into any arrangement which caused Licensor to be unable to enforce his rights under this License.

4. Payment:
In consideration of the License granted under clause 2, Licensee shall pay Licensor a fee, via Credit-Card, PayPal or any other mean which Licensor may deem adequate. Failure to perform payment shall construe as material breach of this Agreement.

5. Upgrades, Updates and Fixes: Licensor may provide Licensee, from time to time, with Upgrades, Updates or Fixes, as detailed herein and according to his sole discretion. Licensee hereby warrants to keep The Software up-to-date and install all relevant updates and fixes, and may, at his sole discretion, purchase upgrades, according to the rates set by Licensor. Licensor shall provide any update or Fix free of charge; however, nothing in this Agreement shall require Licensor to provide Updates or Fixes.

5.1. Upgrades:
for the purpose of this license, an Upgrade shall be a material amendment in The Software, which contains new features and or major performance improvements and shall be marked as a new version number. For example, should Licensee purchase The Software under version 1.X.X, an upgrade shall commence under number 2.0.0.

5.2. Updates:

for the purpose of this license, an update shall be a minor amendment in The Software, which may contain new features or minor improvements and shall be marked as a new sub-version number. For example, should Licensee purchase The Software under version 1.1.X, an upgrade shall commence under number 1.2.0.

5.3. Fix:
for the purpose of this license, a fix shall be a minor amendment in The Software, intended to remove bugs or alter minor features which impair the The Software's functionality. A fix shall be marked as a new sub-sub-version number. For example, should Licensee purchase Software under version 1.1.1, a fix shall commence under number 1.1.2.

6. Support:
Software is provided with limited support, as detailed in the Software's SLA detailed under the License Grant. Licensor shall provide support via the `FooCrypt, A Tale Of Cynical Cyclical Encryption.` issue tracker and / or electronic mail and on regular business days and hours.

6.1. Bug Notification:
Licensee may provide Licensor of details regarding any bug, defect or failure in The Software promptly and with no delay from such event; Licensee shall comply with Licensor's request for information regarding bugs, defects or failures and furnish him with information, screenshots and try to reproduce such bugs, defects or failures.

6.2. Feature Request:
Licensee may request additional features in Software, provided, however, that
(i) Licensee shall waive any claim or right in such feature should feature be developed by Licensor;
(ii) Licensee shall be prohibited from developing the feature, or disclose such feature request, or feature, to any 3rd party directly competing with Licensor or any 3rd party which may be, following the development of such feature, in direct competition with Licensor;
(iii) Licensee warrants that feature does not infringe any 3rd party patent, trademark, trade-secret or any other intellectual property right; and
(iv) Licensee developed, envisioned or created the feature solely by himself.

7. Liability:
To the extent permitted under Law, The Software is provided under an AS-IS basis. Licensor shall never, and without any limit, be liable for any damage, cost, expense or any other payment incurred by Licensee as a result of Software's actions, failure, bugs and/or any other interaction between The Software and Licensee's end-equipment, computers, other software or any 3rd party, end-equipment, computer or services. Moreover, Licensor shall never be liable for any defect in source code written by Licensee when relying on The Software or using The Software's source code.

8. Warranty:

8.1. Intellectual Property:
Licensor hereby warrants that The Software does not violate or infringe any 3rd party claims in regards to intellectual property, patents and/or trademarks and that to the best of its knowledge no legal action has been taken against it for any infringement or violation of any 3rd party intellectual property rights.

8.2. No-Warranty:
The Software is provided without any warranty; Licensor hereby disclaims any warranty that The Software shall be error free, without defects or code which may cause damage to Licensee's computers or to Licensee, and that Software shall be functional. Licensee shall be solely liable to any damage, defect or loss incurred as a result of operating software and undertake the risks contained in running The Software on License's Server[s] and Website[s] and Work Station[s].

8.3. Prior Inspection:
Licensee hereby states that he inspected The Software thoroughly and found it satisfactory and adequate to his needs, that it does not interfere with his regular operation and that it does meet the standards and scope of his computer systems and architecture. Licensee found that The Software interacts with his development, website and server and workstation environment and that it does not infringe any of End User License Agreement of any software Licensee may use in performing his services. Licensee hereby waives any claims regarding The Software's incompatibility, performance, results and features, and warrants that he inspected the The Software.

9. No Refunds:
Licensee warrants that he inspected The Software according to clause 7(c) and that it is adequate to his needs. Accordingly, as The Software is intangible goods, Licensee shall not be, ever, entitled to any refund, rebate, compensation or restitution for any reason whatsoever, even if The Software contains material flaws.

10. Indemnification:
Licensee hereby warrants to hold Licensor harmless and indemnify Licensor for any lawsuit brought against it in regards to Licensee's use of The Software in means that violate, breach or otherwise circumvent this license, Licensor's intellectual property rights or Licensor's title in The Software. Licensor shall promptly notify Licensee in case of such legal action and request Licensee's consent prior to any settlement in relation to such lawsuit or claim.

11. Governing Law, Jurisdiction: Licensee hereby agrees not to initiate class-action lawsuits against Licensor in relation to this license and to compensate Licensor for any legal fees, cost or attorney fees should any claim brought by Licensee against Licensor be denied, in part or in full.

Licensing EULA (End User Licensing Agreement)

End-User License Agreement (EULA) of FooCrypt, A Tale Of Cynical Cyclical Encryption.

This End-User License Agreement (“EULA”) is a legal agreement between you and Cryptopocalypse

This EULA agreement governs your acquisition and use of our FooCrypt, A Tale Of Cynical Cyclical Encryption. software (“Software”) directly from Cryptopocalypse or indirectly through a Cryptopocalypse authorised reseller or distributor (a “Reseller”).

Please read this EULA agreement carefully before completing the installation process and using the FooCrypt, A Tale Of Cynical Cyclical Encryption. software. It provides a license to use the FooCrypt, A Tale Of Cynical Cyclical Encryption. software and contains warranty information and liability disclaimers.

If you register for a free trial of the FooCrypt, A Tale Of Cynical Cyclical Encryption. software, this EULA agreement will also govern that trial. By clicking “accept” or installing and/or using the FooCrypt, A Tale Of Cynical Cyclical Encryption. software, you are confirming your acceptance of the Software and agreeing to become bound by the terms of this EULA agreement.

If you are entering into this EULA agreement on behalf of a company or other legal entity, you represent that you have the authority to bind such entity and its affiliates to these terms and conditions. If you do not have such authority or if you do not agree with the terms and conditions of this EULA agreement, do not install or use the Software, and you must not accept this EULA agreement.

This EULA agreement shall apply only to the Software supplied by Cryptopocalypse herewith regardless of whether other software is referred to or described herein. The terms also apply to any Cryptopocalypse updates, supplements, Internet-based services, and support services for the Software, unless other terms accompany those items on delivery. If so, those terms apply.

License Grant

Cryptopocalypse hereby grants you a personal, non-transferable, non-exclusive licence to use the FooCrypt, A Tale Of Cynical Cyclical Encryption. software on your devices in accordance with the terms of this EULA agreement.

You are permitted to load the FooCrypt, A Tale Of Cynical Cyclical Encryption. software (for example a PC, laptop, mobile or tablet) under your control. You are responsible for ensuring your device meets the minimum requirements of the FooCrypt, A Tale Of Cynical Cyclical Encryption. software.

You are not permitted to:

– Edit, alter, modify, adapt, translate or otherwise change the whole or any part of the Software nor permit the whole or any part of the Software to be combined with or become incorporated in any other software, nor decompile, disassemble or reverse engineer the Software or attempt to do any such things
– Reproduce, copy, distribute, resell or otherwise use the Software for any commercial purpose
– Allow any third party to use the Software on behalf of or for the benefit of any third party
– Use the Software in any way which breaches any applicable local, national or international law
– use the Software for any purpose that Cryptopocalypse considers is a breach of this EULA agreement

Intellectual Property and Ownership

Cryptopocalypse shall at all times retain ownership of the Software as originally downloaded by you and all subsequent downloads of the Software by you. The Software (and the copyright, and other intellectual property rights of whatever nature in the Software, including any modifications made thereto) are and shall remain the property of Cryptopocalypse.

Cryptopocalypse reserves the right to grant licences to use the Software to third parties.

Termination

This EULA agreement is effective from the date you first use the Software and shall continue until terminated. You may terminate it at any time upon written notice to Cryptopocalypse.

It will also terminate immediately if you fail to comply with any term of this EULA agreement. Upon such termination, the licenses granted by this EULA agreement will immediately terminate and you agree to stop all access and use of the Software. The provisions that by their nature continue and survive will survive any termination of this EULA agreement.

Governing Law

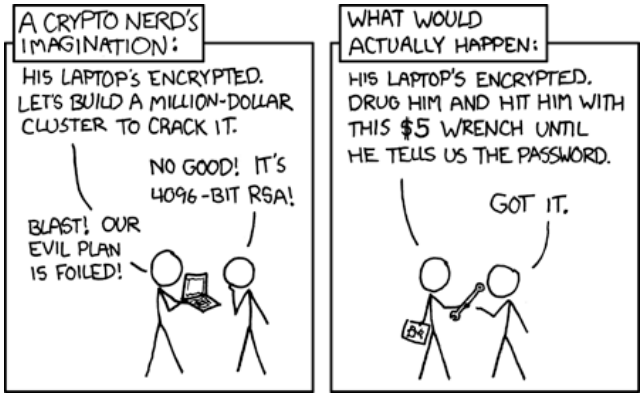
This EULA agreement, and any dispute arising out of or in connection with this EULA agreement, shall be governed by and construed in accordance with the laws of Australia.

Executive Summary

FooCrypt, A Tale Of Cynical Cyclical Encryption is a Post-Quantum Cryptography Quantum+ Proof / Secure software solution via Cryptography and Steganography that provides you with the total peace of mind over the SECURITY & PRIVACY of YOUR DATA, whether it is sitting in situ on a MEDIA DEVICE or in TRANSIT.

Post-Quantum Cryptography (PQC) is about designing cryptographic solutions that can be used by today's [non-quantum] computers and that are resistant to both conventional and quantum cryptanalysis.

There is an age old comic by XKCD regarding the 'Security' of Data, and Brute Force Attacks :



https://imgs.xkcd.com/comics/security.png
https://xkcd.com/538/

FooCrypt, A Tale Of Cynical Cyclical Encryption Makes In-Secure & Secure Symmetric Algorithms Quantum+ Proof / Secure, via Cryptography & Steganography, since May, 2019 via FooCrypt.2.2.0.Core.

With the current emergence of Quantum Computing, the Crypto Nerds imagination has become a reality, one that is expected to occur between 2025 - 2030, because If you encrypt data that needs to be kept confidential for more than 10 years and an adversary could gain access to the encrypted data (cypher text), you need to take action now to protect your data. Otherwise, your security will be compromised as soon as an adversary gains access to a large quantum computer.

The reality of human brute force attacks, is a never ending battle that all those who have access to the passwords which protect the sensitive data and the encrypted data (cypher text), deal with on a daily basis. The advisories may be in person or via the more common electronic means (Hacking Groups / MALWare / Virus's / Worms, Etc).

Resilience is the key to data protection and data security, and utilising a Post-Quantum Cryptography solution to protect and secure your data, guarantees, that in a worst case scenario where your encrypted data is obtained by an adversary, it remains theoretically secured and protected for the longest amount of time.

FooCrypt, A Tale Of Cynical Cyclical Encryption :

- FooCrypt ensures current algorithms have their strength increased to be Quantum+ Proof / Secure.
- Maximises the password length for each encryption cycle.
- Applies <= 200 layers of encryption, per FooKey.
- Maximises the time it would take an adversary to decrypt the encrypted data.
- FooSteg (Steganography) obtains infinite protection of your Data, when the Data Image is in-transit and/or sitting in-situ on a storage device, independent of the Source Image.
- FooStegCypher (Cryptography) enhances the strength of a Data Image when it is in-transit and/or sitting in-situ on a storage device, with the Source Image.
- FooKey_Message (Cryptography and Steganography) enhances the strength of your messaging.
- A FooKey can be created from any commonly sourced file or multiple random data sources (Pseudo and True).
- FooCrypt can be configured so that end users never know the actual password characters utilised to encrypt the data.
- Adheres to Standard Business Security Policies and Implementations.
- Is designed to operate with SMB (CIFS) / NFS / Disk Sharing / Etc , networking technologies.
- Can be transparently integrated with any Business's Document Control / Storage Systems.
- Removes the Common Flaws in all encryption technologies.
- Utilises OpenSSL and/or LibreSSL as the default Encryption Engine.
- Can be configured to utilise any Encryption Engine.
- QRCrypto's eAES® Quantum Resistant Cipher Engine is integrated in FooCrypt.11.0.0.Core, onwards.
- Will be under going Common Criteria EAL2+ and FIPS-140-3 Certifications in 2023 /2024.

References and Recommended further reading :

1. **“Post-Quantum Cryptography: Current state and quantum mitigation”**
   ENSIA (European Union Agency For CyberSecurity), February 2021, and the May 03 2021 update,
   https://www.enisa.europa.eu/publications/post-quantum-cryptography-current-state-and-quantum-mitigation

2. **“What is Quantum Mitigation ????”,**
   A comparative analysis of FooCrypt capabilities verses The ENSIA February, 2021 and the May 03, 2021 ENSIA update.
   Mark A. Lane, Founder, FooCrypt, A Tale Of Cynical Cyclical Encryption, April 21, 2021.
   https://FooCrypt.XYZ/what-is-quantum-mitigation
   https://www.linkedin.com/pulse/what-quantum-mitigation-mark-a-lane/

3. **“Cryptography in a post-quantum world”**
   Accenture Labs, October 4, 2018
   https://www.accenture.com/us-en/insights/technology/quantum-cryptography

4. **“Quantum computing - the time is now”**
   Accenture Labs, June 13, 2017
   https://www.accenture.com/us-en/insights/technology/quantum-computing

* Note :
A current Executive Summary is available @ https://FooCrypt.XYZ

FooCrypt, A Tale Of Cynical Cyclical Encryption

The ‘FooCrypt, A Tale Of Cynical Cyclical Encryption’ Cryptography & Steganography toolkit contains the FooCrypt GUI and FooCrypt CLI binaries (Cryptography) which take ANY cypher, and enhances that cypher’s Key bit strength and makes it more resilient to brute force attacks, by enabling the end user to encrypt / decrypt a message, file by up to 200 layers per runtime, by applying the keys contained inside a FooKey. Which itself is by default protected by a single layer of a cypher.

By default FooCrypt utilised OpenSSL as its cypher engine, selecting AES-256, and utilises a FooKey of 25650 characters, across 50 lines (which creates 50 layers of encryption / decryption), at 512 characters per cycle as per The FooKey Method

Some simple math on the FooCrypt FooKey bit strength / brute force strength is : 4.18640577277337772e+1016 Possible Combinations, with further details available via : https://foocrypt.xyz/brute-strength

Layer 1 : Password Characters : 256 = 2.8935810936531149e+507 Possible Combinations

Layer 1 : Password Characters : 512 = 8.37281154554675569e+1014 Possible Combinations

Layer 50 : Total Password Characters : 25600 = 4.18640577277337772e+1016 Possible Combinations

Layer 200 : Total Password Characters : 102400 = 1.67456230910935062e+1017 Possible Combinations

FooCrypt is also capable of utilising the Quantum Resistant eAES® Cypher Engine by QRCrypto :

Note : QRCrypto's eAES® Quantum Resistant Cipher Engine is integrated in FooCrypt.11.0.0.Core, onwards.

The Strongest, Most efficient PQC on the market

https://QRCrypto.ch

With eAES® at its core, QRC provides the best symmetric and asymmetric quantum-resistant cryptography available, made to work coherently up to a 512-bit symmetric encryption level—double the current industry maximum provided by classical AES encryption. Even at 256-bit level, it's 9 orders of magnitude stronger than standard AES.

How much more robust is QRC's eAES® than AES when faced with a Quantum Computer?			
Security Level	**Gates**	**Depth**	**QUBITS**
AES 256	>2e+152	>2e+145	>2e+12
QRC's eAES® 256	>2e+277	>2e+257	>2e+21

QRC is not just for Quantum

Because it was built to withstand quantum attacks, QRC is incredibly powerful against today's classical attacks.

- QRC protects against side channel attacks, which is unique.
- QRCS protects against ransomware: the data is already strongly encrypted, only the organization holds the key.
- QRC protects against "unknown unknowns" such as ever evolving quantum machine learning-enabled attacks.
- QRC protects against data harvesters collecting data to be decrypted in the future.

Encrypted Fifo Inter Process Communications

'FooCrypt, A Tale Of Cynical Cyclical Encryption', FooCrypt-GUI utilises standard named pipes (FIFO) for one way interprocess communications to share sensitive information :

FooCrypt-GUI -> FIFO -> FooCrypt

FooCrypt-GUI -> FIFO -> FooSteg

The FIFO communications are all encrypted via Tcl/Tk tcllib_1_18+ AES 1.2.1 implementation in Cypher Blocker Chaining (CBC) mode.

The One Time Password (OTP) is initialised upon each occurrence of FooCrypt-GUI, and shared from FooCrypt-GUI into all sub process's : FooCrypt, FooSteg.

The OTP is created by calling the KORN SHELL RANDOM variable, 4 times upon FooCrypt-GUI initialisation. The SHA256 HASH of the OTP is then converted into a 32 byte Binary.

The Initialisation Vector (IV) is created by utilising the FIFO filename as a SHA256 HASH converted into a 16 byte Binary for each FIFO Communication.

The Secret Data is converted to Base64, zero padded to 16 byte blocks, encrypted via Tcl/Tk tcllib_1_18 aes::aes, converted to Base64, sent via the FIFO to FooCrypt or FooSteg, converted from Base64, decrypted via Tcl/Tk tcllib_1_18 aes::aes, converted from Base64.

The **TclAES_Fifo** KORN SHELL with embed Tcl, Secure Fifo Simulation script is provided as part of FooCrypt.X.Y.Z.Core in the Scripts directory for all end users of FooCrypt to explore the method utilised within FooCrypt-GUI, FooCrypt & FooSteg for Encrypted Fifo Inter Process Communications.

FooSteg

‘FooCrypt, A Tale Of Cynical Cyclical Encryption’, FooSteg (Steganography) is able to hide the message / file, inside an image, so that the image can be used to transmit / transit the message / file in plain sight, without any fears or worries, that some adversary, may be able to reconstruct the data inside the image and provide the adversary, with an opportunity to decrypt the message / file, and obtain access to the information.

By default, FooSteg utilises a binary value offset, from a Source Image pixel RGB value, between a min / max RGB value, and writes the binary value of a single character, across 9 individual Red Green & Blue values, to create what FooSteg calls a Data Image. Enhanced obfuscation options to the read / write / extract sequences are performed via the 8 user selected ’Scan Mode’ options, which further enhancements to its obfuscation being available by the user selected Starting Pixel location.

FooSteg utilises 8 different SCAN MODES to create a ScanMap sequence, for applying variations in the WRITE / EXTRACT sequence that applies the binary data into the Data Image.

```
-A [ Scan Mode ]
        * Sequence That FooSteg Scans Pixel RGB Values and Writes / Extracts Binary Data To / From The Images
                * Requires
                  -a [ Analyse | Copy | Extract | Random | Read | Test | Write ]

           * 0 [ Start Scan at top left corner,     Create Scan Map from Top to Bottom, Left to Right  ]
             * Default

           * 1 [ Start Scan at bottom left corner,  Create Scan Map from Bottom to Top, Left to Right ]
           * 2 [ Start Scan at top right corner,    Create Scan Map from Top to Bottom, Right to Left ]
           * 3 [ Start Scan at bottom right corner, Create Scan Map from Bottom to Top, Right to Left ]
           * 4 [ Start Scan at top left corner,     Create Scan Map from Left to Right, Top to Bottom ]
           * 5 [ Start Scan at bottom left corner,  Create Scan Map from Left to Right, Bottom to Top ]
           * 6 [ Start Scan at top right corner,    Create Scan Map from Right to Left, Top to Bottom ]
           * 7 [ Start Scan at bottom right corner, Create Scan Map from Right to Left, Bottom to Top ]


                          0x0 ______________________________________    [ Image Width - 1 ]x0
                              |                                      |
                              |                                      |
                              |                                      |
                              |                                      |
                              |   Any Pixel @ [ Width ]x[ Height ]   |
                              |                                      |
                              |                                      |
                              |                                      |
                              |                                      |
                               ______________________________________
        0x[ Image Height - 1 ]                                    [ Image Width - 1 ]x[ Image Height - 1 ]
```

FooSteg utilises an option for the User to identify a Start Pixel, for applying variations in the WRITE / EXTRACT sequence that applies the binary data into the Data Image.

```
-p [ Starting Pixel [ Width ]x[ Height ] ]
        * Starting Pixel Located @ [Width Pixel]x[Height Pixel] To Be Used By Scan Mode -A [ 0 - 7 ]
                * Default 0x0 [ Top Left Corner Of Image ] For Default Scan Mode -A 0
                * Requires
                  -a [ Analyse | Copy | Extract | Random | Read | Test | Write ]
                * Optional
                  -A [ 0 - 7 ]
                  * 0 [ Start Scan at Pixel [ Width ]x[ Height ], Create Scan Map from Top to Bottom, Left to Right  ]
                    * Default

                  * 1 [ Start Scan at Pixel [ Width ]x[ Height ], Create Scan Map from Bottom to Top, Left to Right ]
                  * 2 [ Start Scan at Pixel [ Width ]x[ Height ], Create Scan Map from Top to Bottom, Right to Left ]
                  * 3 [ Start Scan at Pixel [ Width ]x[ Height ], Create Scan Map from Bottom to Top, Right to Left ]
                  * 4 [ Start Scan at Pixel [ Width ]x[ Height ], Create Scan Map from Left to Right, Top to Bottom ]
                  * 5 [ Start Scan at Pixel [ Width ]x[ Height ], Create Scan Map from Left to Right, Bottom to Top ]
                  * 6 [ Start Scan at Pixel [ Width ]x[ Height ], Create Scan Map from Right to Left, Top to Bottom ]
                  * 7 [ Start Scan at Pixel [ Width ]x[ Height ], Create Scan Map from Right to Left, Bottom to Top ]

                          0x0 ____________________________________ [ Image Width - 1 ]x0
                             |                                    |
                             |                                    |
                             |                                    |
                             |                                    |
                             |   Any Pixel @ [ Width ]x[ Height ]   |
                             |                                    |
                             |                                    |
                             |                                    |
                             |                                    |
                              ____________________________________
        0x[ Image Height - 1 ]                                    [ Image Width - 1 ]x[ Image Height - 1 ]
```

FooSteg utilises an option for the User to identify a RGB Minimum Value (1-253), for applying variations in the WRITE / EXTRACT sequence that applies the binary data into the Data Image.

```
-m [ Min RGB ]
        * <= Minimum RGB Value
        * Default 1
                1-253
                <= Max - 1
```

FooSteg utilises an option for the User to identify a RGB Maximum Value (2-254), for applying variations in the WRITE / EXTRACT sequence that applies the binary data into the Data Image.

```
-M [ Max RGB ]
        * >= Maximum RGB Value
        * Default 254
                2-254
                >= Min + 1
```

FooSteg utilises FooStegCypher via a User supplied KEY (FooStegKey : 8 - 10240 Characters), in addition to a randomly generated 6 digit Token (FooStegToken between 100000 >=< 999999 generated by FooSteg during the Write Mode or User supplied) to reorder the ScanMap, into a repeatable CypherMap, which is utilised as the WriteMap in the Writing the binary data into the Data Image.

```
-k [ FooStegKey Is Asked For Via A Prompt ]
        * Increase The Brute Force Stength Via The FooStegCypher
        * Apply The FooStegKey To The FooStegCypher To Reorganise [ FooStegScanMap -> FooStegCypher -> FooStegWriteMap | FooStegExtractMap ]
        * Minumum Characters 8
        * Maximum Characters 32
        * Creates FooStegToken
        * Optional If Using FooStegKey With Mode
          * Write
          * Extract
          * Test

-K "[ FooStegKey Via A Command Line Option ]" [ Enclosed in Double Quotes ]
        * Increase The Brute Force Stength Via The FooStegCypher
        * Apply The FooStegKey To The FooStegCypher To Reorganise [ FooStegScanMap -> FooStegCypher -> FooStegWriteMap | FooStegExtractMap ]
        * Must Be Enclosed In Double Quotes "[ FooStegKey ]"
        * Minumum Characters 8
        * Maximum Characters 32
        * Creates FooStegToken
        * Optional If Using FooStegKey With Mode
          * Write
          * Extract
          * Test
```

FooSteg utilises FooStegCypher via a User supplied KEY (FooStegKey : 8 - 10240 Characters), in addition to the User supplying a 6 digit Token (FooStegToken between 100000 <=> 999999 randomly generated by FooSteg during the Write Mode or User supplied) to reorder the ScanMap, into a repeatable CypherMap, which is utilised as the ExtractMap in the Extraction the binary data from the Data Image.

```
-t [ FooStegToken Is Asked For Via A Prompt ]
        * Increase The Brute Force Stength Via The FooStegCypher
        * Apply The FooStegToken With The FooStegKey To The FooStegCypher To Reorganise [ FooStegScanMap -> FooStegCypher -> FooStegExtractMap ]
        * Minumum Characters 8
        * Maximum Characters 32
        * Optional If Using FooStegKey With Mode
          * Extract

-T "[ FooStegToken Via A Command Line Option ]" [ Enclosed in Double Quotes ]
        * Increase The Brute Force Stength Via The FooStegCypher
        * Apply The FooStegToken With The FooStegKey To The FooStegCypher To Reorganise [ FooStegScanMap -> FooStegCypher -> FooStegExtractMap ]
        * Must Be Enclosed In Double Quotes "[ FooStegToken ]"
        * Minumum Characters 8
        * Maximum Characters 32
        * Optional If Using FooStegKey With Mode
          * Extract
```

FooSteg enhances the Write & Extract modes binary offsets by enabling the end user to choose between Negative (-N) or Positive (-P) binary offsets, which can also be further enhanced by Bounce Change Oscillations (-B), either at the Pixel for individual RGB layers.

```
-N [ Negative : Negative Binary Offset ]
     * Default : Positive Binary Offset
     * Modes
       * Write
       * Extract

-P [ Positive : Positive Binary Offset ]
     * Default : Positive Binary Offset
     * Modes
       * Write
       * Extract
```

```
-B [ Bounce Change Oscillations : Pixel | RGB ]
     * Default : Pixel
     * Pixel [ Change Oscillations Are Performed On A Per Scanned Pixel Value ]
     * RGB   [ Change Oscillations Are Performed On A Per Scanned Pixel RGB Value ]
     * Modes
       * Write
       * Extract
```

FooSteg Change Numeric (-c), enables the end user to select a value for the change to occur, or to be utilised as input for the Change Formulas.

```
-c [ Change Numeric Value ]
     * Default : 100.0
     * Optional -B [ Bounce Change Oscillations : Pixel | RGB ]
     * Requires -C [ Change Formula ]
     * Minimum 1.0
     * Maximum [ Image Width * Image Height ]
     * Rounded To 5 Decimal Places : 0.12345
     * Modes
       * Copy
         * [ Change Formula Numeric Value ] = ( Percentage Variance Of RGB Values Of The Input Image Intensity )
       * Write
         * [ Change Formula Numeric Value ] = ( Varies Depending On Change Formula )
       * Extract
         * [ Change Formula Numeric Value ] = ( Varies Depending On Change Formula )
       * Test
         * Default : [ Random Number Between 1 - 99 ]
         * [ Change Formula Numeric Value ] = ( Varies Depending On Change Formula )
       * TestVerbose
         * Default : [ Random Number Between 1 - 99 ]
         * [ Change Formula Numeric Value ] = ( Varies Depending On Change Formula )
```

FooSteg Change Formulas, enhance the variations to the Copy, Write & Extract modes.

```
-C [ Change Formula : N | None ,
                      A | Algebraic ,
                      EA | Ecliptic_Area ,
                      EC | Ecliptic_Circumference ,
                      L | Linear ,
                      S1,x,y | Sequence1,x,y ,
                      S2,x,y | Sequence2,x,y ,
                      SW | Sign-Wave ,
                      G | Grayscale ,
                      Neg | Negative ,
                      S,x,y | Sepia,x,y ]
     * Default : None
     * Optional -B [ Bounce Change Oscillations : Pixel | RGB ]
     * Requires -c [ Change Numeric Value ]
```

FooSteg Change formulas (-C), for Copy mode perform a visual change to the source image.

```
* Modes
  * Copy
  * [ N | None , G | Grayscale , Neg | Negative , S,x,y | Sepia,x,y ]

  * N | None
    * [ Change Formula Numeric Value ] = ( Percentage Variance Of RGB Values Of The Input Image Intensity )

  * G | Grayscale
    * [ Change Formula Numeric Value ] = ( Percentage Variance Of RGB Values Of The Input Image Intensity )
                                         * where N = ( [ Change Numeric Value ] / 100 )

  * Neg | Negative
    * [ Change Formula Numeric Value ] = ( Percentage Variance Of RGB Values Of The Input Image Intensity )
                                         * where N = ( [ Change Numeric Value ] / 100 )

  * S,x,y | Sepia,x,y
    * [ Change Formula Numeric Value ] = ( Percentage Variance Of RGB Values Of The Input Image Intensity )
                                         * where N = ( [ Change Numeric Value ] / 100 )
                                         * where x = ( Sepia Depth Value )
                                           * Default 20
                                         * where y = ( Sepia Intensity Value )
                                           * Default 30
```

FooSteg Change formulas (-C) for the Write & Extract modes, enable enhancements to the obfuscation brute strength, by applying different Change Formulas, to the Write & Extract Binary Offset WriteMap & ExtractMap sequences, and are utilised with the Bounce Change Oscillations (-B) (- / +) via Pixel or RGB values per Change Numeric Value (-c), which is modified as per the selected formula.

```
* Modes
  * Write
  * Extract
  * [ N | None , A | Algebraic , EA | Ecliptic_Area , EC | Ecliptic_Circumference , L | Linear , S1,x,y | Sequence1,x,y , S2,x,y | Sequence2,x,y , SW | Sign-Wave ]
  * [ Continually Change From Positive To Negative To Positive ..... Binary Writes / Extracts, Every [ Change Formula Numeric Value ], Till EndRGB / EndBASE64 ]
  * [ Continually Change From Negative To Positive To Negative ..... Binary Writes / Extracts, Every [ Change Formula Numeric Value ], Till EndRGB / EndBASE64 ]

  * N | None

  * A | Algebraic
    * [ Change Formula Numeric Value ] = ( ( r * Prime Number ) / π )
                                         * where r = 100.0
                                         * where Prime Number = Lowest Prime Number Between : rπ and ( rπ+100, rπ+600, rπ+1100, ... )
                                         * where π = 3.14159
                                         * where [ Change Formula Numeric Value ], per Scanned, -B [ Bounce Change Oscillations : Pixel | RGB ] Value

  * EA | Ecliptic_Area
    * [ Change Formula Numeric Value ] = ( πr2 (Pi R Squared)
                                         * where r = [ Change Numeric Value ] )
                                         * where π = 3.14159
                                         * where [ Change Formula Numeric Value ], per Scanned, -B [ Bounce Change Oscillations : Pixel | RGB ] Value

  * EC | Ecliptic_Circumference
    * [ Change Formula Numeric Value ] = ( 2πr ( 2 Pi R )
                                         * where r = [ Change Numeric Value ] )
                                         * where π = 3.14159
                                         * where [ Change Formula Numeric Value ], per Scanned, -B [ Bounce Change Oscillations : Pixel | RGB ] Value

  * L | Linear
    * [ Change Formula Numeric Value ] = ( N )
                                         * where N = [ Change Numeric Value ], per + - or - +
                                         * where [ Change Formula Numeric Value ], per Scanned, -B [ Bounce Change Oscillations : Pixel | RGB ] Value

  * S1,x,y | Sequence1,x,y
    * [ Change Formula Numeric Value ] = ( N,N+(1x),N+(2x),...,N+(yx), .... Repeating )
                                         * Rounded To 5 Decimal Places : 0.12345
                                         * where N = [ Change Numeric Value ] )
                                         * where x = ( Stepped Sequence Value )
                                           * Minimum 0.1
                                         * where y = ( Maximum Sequence Value )
                                           * Minimum 1.0
                                         * where [ Change Formula Numeric Value ], per Scanned, -B [ Bounce Change Oscillations : Pixel | RGB ] Value

  * S2,x,y | Sequence2,x,y
    * [ Change Formula Numeric Value ] = ( N,N+(1x),N+(2x),...,N+(yx),N+(yx)...,N+(2x),N+(1x),N, .... Repeating )
                                         * Rounded To 5 Decimal Places : 0.12345
                                         * where N = [ Change Numeric Value ] )
                                         * where x = ( Stepped Sequence Value )
                                           * Minimum 0.1
                                         * where y = ( Maximum Sequence Value )
                                           * Minimum 1.0
                                         * where [ Change Formula Numeric Value ], per Scanned, -B [ Bounce Change Oscillations : Pixel | RGB ] Value

  * SW | Sign-Wave
    * [ Change Formula Numeric Value ] = ( λ = v / f ( Wave Length = Velocity / Frequency )
                                         * where λ = [ Change Numeric Value ], per + - + or - + -
                                         * where [ Change Formula Numeric Value ], per Scanned, -B [ Bounce Change Oscillations : Pixel | RGB ] Value
```

FooSteg Maximum Obfuscation Brute Strength

When a FooSteg Data Image is In-Transit or resting In-Situ on media but kept apart from the FooSteg Source Image, the Maximum Obfuscation Brute Strength is effectively infinite, as both the Source Image and Data Image are required for any attempt to recover the Secret Data Blob.

Without the Source Image or Data Image(n), to access the Secret Data Blob via FooSteg, an adversary needs to calculate all binary per pixel combinations, due to the options that FooSteg provides to an end user.

FooSteg's Maximum Obfuscation Brute Strength Calculations when the Source and Data Image are present is based on the following FooSteg options :

FooSteg Variation	FooSteg Option	Values
Minimum Pixel Width	-d, -f, -s	10
Minimum Pixel Height	-d, -f, -s	10
Maximum Pixel Width	-d, -f, -s	10000
Maximum Pixel Height	-d, -f, -s	10000
Binary Offsets	-N, -P	2
Change Oscillations	-B	2
Change Formulas	-C	11
Change Numeric Value	-c	1 - N
RGB	-m, -M	3
FooSteg Repeat Write	-r	EndBase64 | EndRGB
Write, Extract Modes	-A	8
Write, Extract Modes Start Pixel	-p	Pixel @ Width x Height
FooSteg Cypher Rounds	-R	19 - 512
FooSteg Cypher FooStegKey	-k, -K	8 - 10240 ASCII Characters

=> A Layer(n) implementation can be utilised on a single Source Image when the Secret Data Blob does not fit within a single layer of a Source Image :

- Layer (1) = [Source Image (+/- Binary Offset) Secret Data Blob(1) = Data Image(1)]
- Layer (2) = [Data Image(1) (+/- Binary Offset) Secret Data Blob(2) = Data Image(2)]
- Layer (n) = [Data Image(n-1) (+/- Binary Offset) Secret Data Blob(n) = Data Image(n)]

The Number of Source and Data Images required to extract the Secret Data Blob can be represented as :

$$=> 2^n : n >= 1, n < \infty$$

=> In-Transit and In-Situ storage of the Source Image and Data Image(n) requires all the Data Image(n), in order to retrieve the complete Secret Data Blob.

=> Where the Source Image or all Data Image(n) are not available to an adversary, the adversary needs to calculate all binary per pixel combinations to extract the Secret Data Blob :

=> State^(Width x Height) :

Width = Number of Pixels Wide
Height = Number of Pixels High
State = 27 Binary States per Pixel RGB Values

000	**001**	**010**	**011**	**100**	**101**	**110**	**111**	**00N**
0N0	**N00**	**NON**	**NN0**	**0NN**	**11N**	**1N1**	**N11**	**N1N**
NN1	**1NN**	**01N**	**N10**	**10N**	**N01**	**0N1**	**1N0**	**NNN**

N = Skip RGB Value, Inserts a 0 into the Binary String (Extract)

FooSteg -m and/or -M Value

Extract Permutation Calculations

∴ **Minimum Image = 27^(10x10)** **=> 1.3689147905e+143**
∴ **Maximum Image = 27^(10000x10000)** **=> 2.6055459178e+143136376**

=> Then the adversary needs to attempt to break any identified cypher text from the 2^n possible Secret Data Blobs identified.

=> FooSteg enables plausible deniability of the Secret Data Blob.

=> When utilised correctly, FooSteg obtains an In-Situ Storage Obfuscation Brute Strength equal to In-Transit of the Data Image.

=> FooCrypt by default uses 50 layers of OpenSSL's AES-256 to enhance its Brute Strength as per The FooKey Method

Layer 1 : Password Characters : 256 = 2.8935810936531149e+507 Possible Combinations

Layer 1 : Password Characters : 512 = 8.37281154554675569e+1014 Possible Combinations

Layer 50 : Total Password Characters : 25600 = 4.18640577277337772e+1016 Possible Combinations

Layer 200 : Total Password Characters : 102400 = 1.67456230910935062e+1017 Possible Combinations

=> FooCrypt is also capable of utilising the Quantum Resistant eAES® Cypher Engine by QRCrypto :

Note : QRCrypto's eAES® Quantum Resistant Cipher Engine is integrated in FooCrypt.11.0.0.Core, onwards.

The Strongest, Most efficient PQC on the market

https://QRCrypto.ch

With eAES® at its core, QRC provides the best symmetric and asymmetric quantum-resistant cryptography available, made to work coherently up to a 512-bit symmetric encryption level—double the current industry maximum provided by classical AES encryption. Even at 256-bit level, it's 9 orders of magnitude stronger than standard AES.

How much more robust is QRC's eAES® than AES when faced with a Quantum Computer?			
Security Level	**Gates**	**Depth**	**QUBITS**
AES 256	>2e+152	>2e+145	>2e+12
QRC's eAES® 256	>2e+277	>2e+257	>2e+21

QRC is not just for Quantum

Because it was built to withstand quantum attacks, QRC is incredibly powerful against today's classical attacks.

- QRC protects against side channel attacks, which is unique.
- QRCS protects against ransomware: the data is already strongly encrypted, only the organization holds the key.
- QRC protects against "unknown unknowns" such as ever evolving quantum machine learning-enabled attacks.
- QRC protects against data harvesters collecting data to be decrypted in the future.

Extract Permutation Calculations

The Extract Permutation Brute Force Calculations are based on a single permutation length, and not the sum of all combinations of permutation lengths, hence the values below are considerably smaller than the actual Extract Permutation Brute Force Calculation Values.

- A permutation is an arrangement of a set of objects in a specific order. In this case, we are talking about permutations of binary values in a pixel. For example, if we have a pixel with three binary values {0, 1, N}, without repetitions, the permutations of that pixel would be:

  01N, 0N1, 10N, 1N0, N01, N10.

- Repetitions refer to the number of times an item appears in a sequence or a set. In this case, repetitions of binary values in a sequence are allowed. For example, a sequence with the values 000 or 11N or NN0 would be allowed.

000	**001**	**010**	**011**	**100**	**101**	**110**	**111**	**00N**
0N0	**N00**	**NON**	**NN0**	**0NN**	**11N**	**1N1**	**N11**	**N1N**
NN1	**1NN**	**01N**	**N10**	**10N**	**N01**	**0N1**	**1N0**	**NNN**

N = Skip RGB Value, Inserts a 0 into the Extract Binary String
FooSteg -m and/or -M Value

- To find all possible permutations of binary values in a pixel, you can follow this step:

  Take the number of binary values in the pixel (27) and raise it to the number of pixels.

∴ **27^(10x10)** **=> 1.3689147905e+143**
∴ **27^(10000x10000)** **=> 2.6055459178e+143136376**

Power of 2

Converting 27^(Width*Height) to a power of 2 using logarithmic properties

We can use the following formula to convert a number from base a to base b:

• log_b(x) = log_a(x) / log_a(b)

Using this formula, we can convert 27 to base 2 as follows:

• log2(27) = log10(27) / log10(2)

Then, we can use the power rule of logarithms to simplify the exponent (10 * 10) :

• 27^(10*10) = (2^(log2(27)))^(10*10) = 2^(log2(27) * 10*10)

Substituting the value of log2(27) from the above calculation :

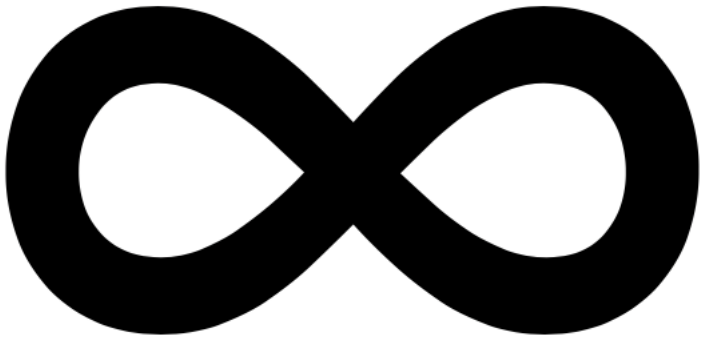
• 27^(1010) = 2^(log10(27) / log10(2) * 1010)

∴ **27^(10x10) => 2e+(146.849)**
∴ **27^(10000x10000) => 2e+(4.754888e+24)**

FooCrypt when combined with FooSteg :

‘FooCrypt makes a mountain out of a mole hill, and FooSteg hides the mountain in plain sight’

Protecting Data via An Effectively Infinite Brute Force & Obfuscation Solution(1)



(1) Lay persons terms representation of calculations covering combinations of a Data Image to recover the Secret Data Blob. Infinity symbol utilised to represent compute power verses time, utilising known side channel attacks methods, brute force attacks methods, et al.

Binary RGB Encoding & Decoding

FooSteg performs Steganography via Binary RGB Encoding & Decoding Of A Base64 File Into & From An Image.

- FooSteg Supports The Following Image Formats / Functionality.

  - GIF & JPEG Formats Utilise A Compression Algorithm Which Prevents The Format From Being The Data Carrier For The Binary RGB Encoding / Decoding

    ID = Input Data Image [See -d & -D]
    IF = Input File Image [See -f & -F]
    IS = Input Source Image [See -s & -S]

    OC = Output Copy Image [See -o & -O]
    OD = Output Data Image [See -o & -O]
    OR = Output Random Image [See -o & -O]

    NO = Image Format Not Supported

Format	Copy	Extract	Random	Read	Write
BMP	IF OC	IS ID	OR	IF	IF OD
GIF	IF	IS	NO	IF	IF
JPEG	IF OC	IS	OR	IF	IF
PCX	IF OC	IS ID	OR	IF	IF OD
PNG	IF OC	IS ID	OR	IF	IF OD
PPM	IF OC	IS ID	OR	IF	IF OD
SGI	IF OC	IS ID	OR	IF	IF OD
SUN	IF OC	IS ID	OR	IF	IF OD
TGA	IF OC	IS ID	OR	IF	IF OD
TIFF	IF OC	IS ID	OR	IF	IF OD

Successful BASE64 Steganography Image Encode / Decode Table	
Input Source Image Format	Data Source Image Format
BMP	BMP, PCX, PNG, PPM, SGI, SUN, TGA, TIFF
GIF	BMP, PCX, PNG, PPM, SGI, SUN, TGA, TIFF
JPEG	BMP, PCX, PNG, PPM, SGI, SUN, TGA, TIFF
PCX	BMP, PCX, PNG, PPM, SGI, SUN, TGA, TIFF
PNG	BMP, PCX, PNG, PPM, SGI, SUN, TGA, TIFF
PPM	BMP, PCX, PNG, PPM, SGI, SUN, TGA, TIFF
SGI	BMP, PCX, PNG, PPM, SGI, SUN, TGA, TIFF
SUN	BMP, PCX, PNG, PPM, SGI, SUN, TGA, TIFF
TGA	BMP, PCX, PNG, PPM, SGI, SUN, TGA, TIFF
TIFF	BMP, PCX, PNG, PPM, SGI, SUN, TGA, TIFF

Binary RGB Pixel Read and Write Routines

- FooSteg currently performs a pixel by pixel read / write of an image as per the ScanMap created via the -A [0 - 7] switch in association with the -p switch to select a non default starting pixel location by suppling a [Pixel Width]x[Pixel Height]

```
-A [ Scan Mode ]
        * Sequence That FooSteg Scans Pixel RGB Values and Writes / Extracts Binary Data To / From The Images
                * Requires
                  -a [ Analyse | Copy | Extract | Random | Read | Test | Write ]

           * 0 [ Start Scan at top left corner,     Create Scan Map from Top to Bottom, Left to Right  ]
                * Default

           * 1 [ Start Scan at bottom left corner,  Create Scan Map from Bottom to Top, Left to Right ]
           * 2 [ Start Scan at top right corner,    Create Scan Map from Top to Bottom, Right to Left ]
           * 3 [ Start Scan at bottom right corner, Create Scan Map from Bottom to Top, Right to Left ]
           * 4 [ Start Scan at top left corner,     Create Scan Map from Left to Right, Top to Bottom ]
           * 5 [ Start Scan at bottom left corner,  Create Scan Map from Left to Right, Bottom to Top ]
           * 6 [ Start Scan at top right corner,    Create Scan Map from Right to Left, Top to Bottom ]
           * 7 [ Start Scan at bottom right corner, Create Scan Map from Right to Left, Bottom to Top ]


                          0x0 ______________________________________     [ Image Width - 1 ]x0
                              |                                      |
                              |                                      |
                              |                                      |
                              |                                      |
                              |   Any Pixel @ [ Width ]x[ Height ]   |
                              |                                      |
                              |                                      |
                              |                                      |
                              |                                      |
                               ______________________________________
        0x[ Image Height - 1 ]                                       [ Image Width - 1 ]x[ Image Height - 1 ]
```

```
-p [ Starting Pixel [ Width ]x[ Height ] ]
        * Starting Pixel Located @ [Width Pixel]x[Height Pixel] To Be Used By Scan Mode -A [ 0 - 7 ]
                * Default 0x0 [ Top Left Corner Of Image ] For Default Scan Mode -A 0
                * Requires
                  -a [ Analyse | Copy | Extract | Random | Read | Test | Write ]
                * Optional
                  -A [ 0 - 7 ]
           * 0 [ Start Scan at Pixel [ Width ]x[ Height ], Create Scan Map from Top to Bottom, Left to Right  ]
                * Default

           * 1 [ Start Scan at Pixel [ Width ]x[ Height ], Create Scan Map from Bottom to Top, Left to Right ]
           * 2 [ Start Scan at Pixel [ Width ]x[ Height ], Create Scan Map from Top to Bottom, Right to Left ]
           * 3 [ Start Scan at Pixel [ Width ]x[ Height ], Create Scan Map from Bottom to Top, Right to Left ]
           * 4 [ Start Scan at Pixel [ Width ]x[ Height ], Create Scan Map from Left to Right, Top to Bottom ]
           * 5 [ Start Scan at Pixel [ Width ]x[ Height ], Create Scan Map from Left to Right, Bottom to Top ]
           * 6 [ Start Scan at Pixel [ Width ]x[ Height ], Create Scan Map from Right to Left, Top to Bottom ]
           * 7 [ Start Scan at Pixel [ Width ]x[ Height ], Create Scan Map from Right to Left, Bottom to Top ]

                          0x0 ______________________________________ [ Image Width - 1 ]x0
                              |                                      |
                              |                                      |
                              |                                      |
                              |                                      |
                              |   Any Pixel @ [ Width ]x[ Height ]   |
                              |                                      |
                              |                                      |
                              |                                      |
                              |                                      |
                               ______________________________________
        0x[ Image Height - 1 ]                                       [ Image Width - 1 ]x[ Image Height - 1 ]
```

- The RGB values for each pixel are tested against the RGB Minimum and RGB Maximum values specified in the FooSteg Preferences.
- If an RGB value is equal to or greater than the RGB Minimum and equal to or less than the RGB Maximum, the RGB value is increased by the binary value of the ASCII character, spread across a set of 9 RGB values.
  - ie for -a Write :
    - ASCII Character 65 = 'A' = 1000001 in Binary. The binary value 1000001 is then padded so that it contains 9 characters by adding leading zeros which produces ‘001000001’. FooSteg writes the values as an offset from the current RGB value of a pixel. So if the pixels located at 0.0 through 0.2 contains RGB values of 10 100 10, and the RGB Min / Max settings are 1 254 respectively, FooSteg will write the pixels located at 0.0 through 0.2 with their new RGB values of 10 100 11, 10 100 10, 10 100 11. This is repeated until all characters are written to the Data Image.

- **Example Verbose : Write : FooSteg -a Write -v Write1 -V**

```
VERBOSE : Write1     : Pixel @ 20 x 3                : Source RGB = 192   59  193 : Binary = '0' '0' '1' : Data RGB = 192   59  194
VERBOSE : Write1     : Pixel @ 20 x 6                : Source RGB =  46    0  103 : Binary = '1' 'X' '0' : Data RGB =  47    0  103
VERBOSE : Write1     : Pixel @ 20 x 15               : Source RGB = 146  114  207 : Binary = '1' '0' '0' : Data RGB = 147  114  207
VERBOSE : Write1     : Pixel @ 20 x 16               : Source RGB = 108  153  162 : Binary = '1' '0' '0' : Data RGB = 109  153  162
VERBOSE : Write1     : Pixel @ 20 x 33               : Source RGB = 186   99   15 : Binary = '1' '0' '1' : Data RGB = 187   99   16
VERBOSE : Write1     : Pixel @ 20 x 36               : Source RGB = 235  229  129 : Binary = '0' '1' '1' : Data RGB = 235  230  130
VERBOSE : Write1     : Pixel @ 2 x 4                 : Source RGB =  26  239  155 : Binary = '0' '0' '0' : Data RGB =  26  239  155
VERBOSE : Write1     : Pixel @ 21 x 2                : Source RGB = 195  141  173 : Binary = '1' '0' '0' : Data RGB = 196  141  173
VERBOSE : Write1     : Pixel @ 21 x 5                : Source RGB =  15  244   93 : Binary = '0' '0' '1' : Data RGB =  15  244   92
VERBOSE : Write1     : Pixel @ 2 x 6                 : Source RGB = 142   48  255 : Binary = '0' '0' 'X' : Data RGB = 142   48  255
VERBOSE : Write1     : Pixel @ 2 x 7                 : Source RGB =  74  129   12 : Binary = '0' '1' '0' : Data RGB =  74  128   12
VERBOSE : Write1     : Pixel @ 21 x 21               : Source RGB = 108   83  171 : Binary = '0' '1' '1' : Data RGB = 108   82  170
VERBOSE : Write1     : Pixel @ 21 x 25               : Source RGB = 133   50  223 : Binary = '1' '1' '0' : Data RGB = 132   49  223
VERBOSE : Write1     : Pixel @ 2 x 8                 : Source RGB =  61  188   93 : Binary = '0' '1' '0' : Data RGB =  61  187   93
VERBOSE : Write1     : Pixel @ 21 x 30               : Source RGB =  37   72   15 : Binary = '1' '0' '0' : Data RGB =  36   72   15
VERBOSE : Write1     : Pixel @ 21 x 40               : Source RGB =  63   44    7 : Binary = '1' '0' '0' : Data RGB =  62   44    7
VERBOSE : Write1     : Pixel @ 21 x 46               : Source RGB = 116  127  209 : Binary = '0' '1' '1' : Data RGB = 116  126  210
```

- **Example Verbose : Write : FooSteg -a Write -v Write2 -V**

```
VERBOSE : Write2     : Binary Character = 1          : Positive Write Binary Character '0' To Pixel @ 20 x 3 : 192  ->  192 : Red
VERBOSE : Write2     : Binary Character = 2          : Positive Write Binary Character '0' To Pixel @ 20 x 3 :  59  ->   59 : Green
VERBOSE : Write2     : Binary Character = 3          : Write Binary Character '1' To Pixel @ 20 x 3 : 193  ->  194 : Blue
VERBOSE : Write2     : Binary Character = 4          : Positive Write Binary Character '1' To Pixel @ 20 x 6 :  46  ->   47 : Red
VERBOSE : Write2     : Binary Character = 5          : Write Binary Character '0' To Pixel @ 20 x 6 : 103  ->  103 : Blue
VERBOSE : Write2     : Binary Character = 6          : Positive Write Binary Character '1' To Pixel @ 20 x 15 : 146  ->  147 : Red
VERBOSE : Write2     : Binary Character = 7          : Positive Write Binary Character '0' To Pixel @ 20 x 15 : 114  ->  114 : Green
VERBOSE : Write2     : Binary Character = 8          : Write Binary Character '0' To Pixel @ 20 x 15 : 207  ->  207 : Blue
VERBOSE : Write2     : Binary Character = 9          : Positive Write Binary Character '1' To Pixel @ 20 x 16 : 108  ->  109 : Red
VERBOSE : Write2     : Binary Character = 10         : Positive Write Binary Character '0' To Pixel @ 20 x 16 : 153  ->  153 : Green
VERBOSE : Write2     : Binary Character = 11         : Write Binary Character '0' To Pixel @ 20 x 16 : 162  ->  162 : Blue
VERBOSE : Write2     : Binary Character = 12         : Positive Write Binary Character '1' To Pixel @ 20 x 33 : 186  ->  187 : Red
VERBOSE : Write2     : Binary Character = 13         : Positive Write Binary Character '0' To Pixel @ 20 x 33 :  99  ->   99 : Green
VERBOSE : Write2     : Binary Character = 14         : Write Binary Character '1' To Pixel @ 20 x 33 :  15  ->   16 : Blue
VERBOSE : Write2     : Binary Character = 15         : Positive Write Binary Character '0' To Pixel @ 20 x 36 : 235  ->  235 : Red
VERBOSE : Write2     : Binary Character = 16         : Positive Write Binary Character '1' To Pixel @ 20 x 36 : 229  ->  230 : Green
VERBOSE : Write2     : Binary Character = 17         : Write Binary Character '1' To Pixel @ 20 x 36 : 129  ->  130 : Blue
VERBOSE : Write2     : Binary Character = 18         : Positive Write Binary Character '0' To Pixel @ 2 x 4 :  26  ->   26 : Red
VERBOSE : Write2     : Binary Character = 19         : Positive Write Binary Character '0' To Pixel @ 2 x 4 : 239  ->  239 : Green
VERBOSE : Write2     : Binary Character = 20         : Write Binary Character '0' To Pixel @ 2 x 4 : 155  ->  155 : Blue
VERBOSE : Write2     : Binary Character = 21         : Positive Write Binary Character '1' To Pixel @ 21 x 2 : 195  ->  196 : Red
VERBOSE : Write2     : Binary Character = 22         : Positive Write Binary Character '0' To Pixel @ 21 x 2 : 141  ->  141 : Green
VERBOSE : Write2     : Binary Character = 23         : Write Binary Character '0' To Pixel @ 21 x 2 : 173  ->  173 : Blue
VERBOSE : Write2     : Binary Character = 24         : Positive Write Binary Character '0' To Pixel @ 21 x 5 :  15  ->   15 : Red
VERBOSE : Write2     : Binary Character = 25         : Negative Write Binary Character '0' To Pixel @ 21 x 5 : 244  ->  244 : Green
VERBOSE : Write2     : Binary Character = 26         : Write Binary Character '1' To Pixel @ 21 x 5 :  93  ->   92 : Blue
VERBOSE : Write2     : Binary Character = 27         : Negative Write Binary Character '0' To Pixel @ 2 x 6 : 142  ->  142 : Red
VERBOSE : Write2     : Binary Character = 28         : Negative Write Binary Character '0' To Pixel @ 2 x 6 :  48  ->   48 : Green
VERBOSE : Write2     : Binary Character = 29         : Negative Write Binary Character '0' To Pixel @ 2 x 7 :  74  ->   74 : Red
VERBOSE : Write2     : Binary Character = 30         : Negative Write Binary Character '1' To Pixel @ 2 x 7 : 129  ->  128 : Green
VERBOSE : Write2     : Binary Character = 31         : Write Binary Character '0' To Pixel @ 2 x 7 :  12  ->   12 : Blue
```

- The base64 data is then extracted from the Data Image, using the Source Image, to recreate the base64 file for comparison with the input base64 file.

- **Example Verbose StdOut : Extract Test : FooSteg -a Write -v Extract1 -V**

```
VERBOSE : Extract1   : Pixel @ 20 x 3          : Source RGB = 192   59  193 : Data RGB = 192   59  194 : Binary = '0'  '0'  '1'
VERBOSE : Extract1   : Pixel @ 20 x 6          : Source RGB =  46    0  103 : Data RGB =  47    0  103 : Binary = '1'  'X'  '0'
VERBOSE : Extract1   : Pixel @ 20 x 15         : Source RGB = 146  114  207 : Data RGB = 147  114  207 : Binary = '1'  '0'  '0'
VERBOSE : Extract1   : Pixel @ 20 x 16         : Source RGB = 108  153  162 : Data RGB = 109  153  162 : Binary = '1'  '0'  '0'
VERBOSE : Extract1   : Pixel @ 20 x 33         : Source RGB = 186   99   15 : Data RGB = 187   99   16 : Binary = '1'  '0'  '1'
VERBOSE : Extract1   : Pixel @ 20 x 36         : Source RGB = 235  229  129 : Data RGB = 235  230  130 : Binary = '0'  '1'  '1'
VERBOSE : Extract1   : Pixel @ 2 x 4           : Source RGB =  26  239  155 : Data RGB =  26  239  155 : Binary = '0'  '0'  '0'
VERBOSE : Extract1   : Pixel @ 21 x 2          : Source RGB = 195  141  173 : Data RGB = 196  141  173 : Binary = '1'  '0'  '0'
VERBOSE : Extract1   : Pixel @ 21 x 5          : Source RGB =  15  244   93 : Data RGB =  15  244   92 : Binary = '0'  '0'  '1'
VERBOSE : Extract1   : Pixel @ 2 x 6           : Source RGB = 142   48  255 : Data RGB = 142   48  255 : Binary = '0'  '0'  'X'
VERBOSE : Extract1   : Pixel @ 2 x 7           : Source RGB =  74  129   12 : Data RGB =  74  128   12 : Binary = '0'  '1'  '0'
VERBOSE : Extract1   : Pixel @ 21 x 21         : Source RGB = 108   83  171 : Data RGB = 108   82  170 : Binary = '0'  '1'  '1'
VERBOSE : Extract1   : Pixel @ 21 x 25         : Source RGB = 133   50  223 : Data RGB = 132   49  223 : Binary = '1'  '1'  '0'
VERBOSE : Extract1   : Pixel @ 2 x 8           : Source RGB =  61  188   93 : Data RGB =  61  187   93 : Binary = '0'  '1'  '0'
VERBOSE : Extract1   : Pixel @ 21 x 30         : Source RGB =  37   72   15 : Data RGB =  36   72   15 : Binary = '1'  '0'  '0'
VERBOSE : Extract1   : Pixel @ 21 x 40         : Source RGB =  63   44    7 : Data RGB =  62   44    7 : Binary = '1'  '0'  '0'
VERBOSE : Extract1   : Pixel @ 21 x 46         : Source RGB = 116  127  209 : Data RGB = 116  126  210 : Binary = '0'  '1'  '1'
VERBOSE : Extract1   : Pixel @ 22 x 6          : Source RGB =  27  152  170 : Data RGB =  28  152  171 : Binary = '1'  '0'  '1'
VERBOSE : Extract1   : Pixel @ 22 x 15         : Source RGB =  72  171   40 : Data RGB =  73  172   40 : Binary = '1'  '1'  '0'
VERBOSE : Extract1   : Pixel @ 22 x 20         : Source RGB = 161   53   15 : Data RGB = 161   53   16 : Binary = '0'  '0'  '1'
VERBOSE : Extract1   : Pixel @ 2 x 13          : Source RGB =  14   20  222 : Data RGB =  15   20  222 : Binary = '1'  '0'  '0'
VERBOSE : Extract1   : Pixel @ 22 x 36         : Source RGB = 116  143  214 : Data RGB = 116  143  214 : Binary = '0'  '0'  '0'
VERBOSE : Extract1   : Pixel @ 22 x 37         : Source RGB = 172  108  159 : Data RGB = 172  109  159 : Binary = '0'  '1'  '0'
```

- **Example Verbose StdOut : Extract Test : FooSteg -a Write -v Extract2 -V**

```
VERBOSE : Extract2   : Pixel @ 0 x 0           : Source Red = 147 : Data Red = 147 : Positive Binary = '0'
VERBOSE : Extract2   : Pixel @ 0 x 0           : Source Blue = 124 : Data Blue = 124 : Positive Binary = '0'
VERBOSE : Extract2   : Pixel @ 0 x 1           : Source Green = 165 : Data Green = 165 : Positive Binary = '0'
VERBOSE : Extract2   : Pixel @ 0 x 2           : Source Red = 150 : Data Red = 151 : Positive Binary = '1'
VERBOSE : Extract2   : Pixel @ 0 x 2           : Source Green = 230 : Data Green = 231 : Positive Binary = '1'
VERBOSE : Extract2   : Pixel @ 0 x 2           : Source Blue = 184 : Data Blue = 185 : Positive Binary = '1'
VERBOSE : Extract2   : Pixel @ 0 x 3           : Source Green = 218 : Data Green = 218 : Positive Binary = '0'
VERBOSE : Extract2   : Pixel @ 0 x 4           : Source Red = 227 : Data Red = 227 : Positive Binary = '0'
VERBOSE : Extract2   : Pixel @ 0 x 4           : Source Green = 224 : Data Green = 224 : Positive Binary = '0'
VERBOSE : Extract2   : Pixel @ 0 x 4           : Source Blue = 160 : Data Blue = 160 : Positive Binary = '0'
VERBOSE : Extract2   : Pixel @ 0 x 5           : Source Red = 224 : Data Red = 224 : Positive Binary = '0'
VERBOSE : Extract2   : Pixel @ 0 x 5           : Source Green = 224 : Data Green = 225 : Positive Binary = '1'
VERBOSE : Extract2   : Pixel @ 0 x 5           : Source Blue = 186 : Data Blue = 187 : Positive Binary = '1'
VERBOSE : Extract2   : Pixel @ 0 x 7           : Source Blue = 138 : Data Blue = 139 : Positive Binary = '1'
VERBOSE : Extract2   : Pixel @ 0 x 8           : Source Red = 197 : Data Red = 198 : Positive Binary = '1'
VERBOSE : Extract2   : Pixel @ 0 x 10          : Source Green = 140 : Data Green = 140 : Positive Binary = '0'
VERBOSE : Extract2   : Pixel @ 0 x 10          : Source Blue = 147 : Data Blue = 147 : Positive Binary = '0'
VERBOSE : Extract2   : Extracted Binary Data   : 001101001
VERBOSE : Extract2   : Extracted Binary Data   : 001010110
VERBOSE : Extract2   : Extracted Binary Data   : 001000010
VERBOSE : Extract2   : Extracted Binary Data   : 001001111
VERBOSE : Extract2   : Extracted Binary Data   : 001010010
VERBOSE : Extract2   : Extracted Binary Data   : 001110111
VERBOSE : Extract2   : Extracted Binary Data   : 000110000
VERBOSE : Extract2   : Extracted Binary Data   : 001001011
VERBOSE : Extract2   : Extracted Binary Data   : 001000111
VERBOSE : Extract2   : Extracted Binary Data   : 001100111
VERBOSE : Extract2   : Extracted Binary Data   : 001101111
VERBOSE : Extract2   : Extracted Binary Data   : 001000001
VERBOSE : Extract2   : Extracted Binary Data   : 001000001
VERBOSE : Extract2   : Extracted Binary Data   : 001000001
VERBOSE : Extract2   : Extracted Binary Data   : 001000001
VERBOSE : Extract2   : Extracted Binary Data   : 001001110
VERBOSE : Extract2   : Extracted Binary Data   : 001010011
```

- ie for -a Extract :
  - ASCII Character 65 = 'A' = 1000001 in Binary. The binary value 1000001 is then padded so that it contains 9 characters by adding leading zeros which produces ‘001000001’. FooSteg reads the RGB values of the Source Image and the Data Image comparing each RGB value for each pixel location based on the RGB Minimum and RGB Maximum settings.

  - So if the pixels in the Source Image located at 0.0 through 0.2 contains RGB values of 10 100 10, and the RGB Min / Max settings are 1 254 respectively and the pixels in the Data Image located at 0.0 through 0.2 contains RGB values of 10 100 11, 10 100 10, 10 100 11, the comparison of the binary values produces the binary number ‘001000001’ which is then converted to its ASCII Character value ‘A’. This process is continued by reading and comparing each pixels RGB values until FooSteg identifies the EOF marker by extracting 9 zeros during comparison of the RGB pixel values. The base64 formatted file is then created.

- **Example Verbose StdOut : Extract : FooSteg -a Extract -v ExtractMap,Extract1,Extract2 -V**

```
STATUS  : Extract    : Extracting Binary Data                    :
STATUS  : Extract    : Data Image File Name                      : /Users/foocrypt/Data.TIFF
STATUS  : Extract    : Data Image Width                          : 50
STATUS  : Extract    : Data Image Height                         : 50
STATUS  : Extract    : Data Image Frames                         : 1
STATUS  : Extract    : Data Image Total Pixels                   : 2500
STATUS  : Extract    :
STATUS  : Extract    : Src Image File Name                       : /Users/foocrypt/Source.TIFF
STATUS  : Extract    : Src Image Width                           : 50
STATUS  : Extract    : Src Image Height                          : 50
STATUS  : Extract    : Src Image Frames                          : 1
STATUS  : Extract    : Src Image Total Pixels                    : 2500
STATUS  : Extract    :
STATUS  : Extract    : Min RGB                                   : 123
STATUS  : Extract    : Max RGB                                   : 231
STATUS  : Extract    :
VERBOSE :
VERBOSE : ExtractMap :                                           :
VERBOSE : ExtractMap : Generating ExtractMap Logs                : /Users/foocrypt/Extract.log
VERBOSE : ExtractMap :                                           :
VERBOSE :
VERBOSE : ExtractMap :                                           : Pixel 0 @ 0x0
VERBOSE : ExtractMap :                                           : Pixel 1 @ 0x1
VERBOSE : ExtractMap :                                           : Pixel 2 @ 0x2
VERBOSE : ExtractMap :                                           : Pixel 3 @ 0x3
VERBOSE : ExtractMap :                                           : Pixel 4 @ 0x4
VERBOSE : ExtractMap :                                           : Pixel 5 @ 0x5
VERBOSE : ExtractMap :                                           : Pixel 6 @ 0x6
VERBOSE : ExtractMap :                                           : Pixel 7 @ 0x7
VERBOSE : ExtractMap :                                           : Pixel 8 @ 0x8
VERBOSE : ExtractMap :                                           : Pixel 9 @ 0x9
VERBOSE : ExtractMap :                                           : Pixel 10 @ 0x10
```

<-- CUT -->

```
VERBOSE : Extract1   :                                           :
VERBOSE : Extract1   : Generating Extract1 Logs                  : /Users/foocrypt/Extract.log
VERBOSE : Extract1   :                                           :
VERBOSE :
STATUS  : Extract    : Processed Percent                         : 0%
STATUS  : Extract    : Processed Pixels                          : 0
STATUS  : Extract    : Elapsed Time                              : 596 Milliseconds
STATUS  : Extract    :
VERBOSE : Extract2   : Pixel @ 0 x 0                             : Source Red = 147 : Data Red = 147 : Positive Binary = '0'
VERBOSE : Extract2   : Pixel @ 0 x 0                             : Source Blue = 124 : Data Blue = 124 : Positive Binary = '0'
VERBOSE : Extract1   : Pixel @ 0 x 0                             : Source RGB = 147   73  124 : Data RGB = 147   73  124 : Positive Binary = '0'   'X'   '0'
VERBOSE : Extract2   : Pixel @ 0 x 1                             : Source Green = 165 : Data Green = 165 : Positive Binary = '0'
VERBOSE : Extract1   : Pixel @ 0 x 1                             : Source RGB =  49   165    68 : Data RGB =  49   165    68 : Positive Binary = 'X'   '0'   'X'
VERBOSE : Extract2   : Pixel @ 0 x 2                             : Source Red = 150 : Data Red = 151 : Positive Binary = '1'
VERBOSE : Extract2   : Pixel @ 0 x 2                             : Source Green = 230 : Data Green = 231 : Positive Binary = '1'
VERBOSE : Extract2   : Pixel @ 0 x 2                             : Source Blue = 184 : Data Blue = 185 : Positive Binary = '1'
VERBOSE : Extract1   : Pixel @ 0 x 2                             : Source RGB = 150   230   184 : Data RGB = 151   231   185 : Positive Binary = '1'   '1'   '1'
VERBOSE : Extract2   : Pixel @ 0 x 3                             : Source Green = 218 : Data Green = 218 : Positive Binary = '0'
VERBOSE : Extract1   : Pixel @ 0 x 3                             : Source RGB =   7   218    24 : Data RGB =   7   218    24 : Positive Binary = 'X'   '0'   'X'
VERBOSE : Extract2   : Pixel @ 0 x 4                             : Source Red = 227 : Data Red = 227 : Positive Binary = '0'
VERBOSE : Extract2   : Pixel @ 0 x 4                             : Source Green = 224 : Data Green = 224 : Positive Binary = '0'
VERBOSE : Extract2   : Pixel @ 0 x 4                             : Source Blue = 160 : Data Blue = 160 : Positive Binary = '0'
VERBOSE : Extract1   : Pixel @ 0 x 4                             : Source RGB = 227   224   160 : Data RGB = 227   224   160 : Positive Binary = '0'   '0'   '0'
VERBOSE : Extract2   : Pixel @ 0 x 5                             : Source Red = 224 : Data Red = 224 : Positive Binary = '0'
VERBOSE : Extract2   : Pixel @ 0 x 5                             : Source Green = 224 : Data Green = 225 : Positive Binary = '1'
VERBOSE : Extract2   : Pixel @ 0 x 5                             : Source Blue = 186 : Data Blue = 187 : Positive Binary = '1'
VERBOSE : Extract1   : Pixel @ 0 x 5                             : Source RGB = 224   224   186 : Data RGB = 224   225   187 : Positive Binary = '0'   '1'   '1'
VERBOSE : Extract1   : Pixel @ 0 x 6                             : Source RGB = 234   121   242 : Data RGB = 234   121   242 : Positive Binary = 'X'   'X'   'X'
VERBOSE : Extract2   : Pixel @ 0 x 7                             : Source Blue = 138 : Data Blue = 139 : Positive Binary = '1'
VERBOSE : Extract1   : Pixel @ 0 x 7                             : Source RGB =  37   238   138 : Data RGB =  37   238   139 : Positive Binary = 'X'   'X'   '1'
VERBOSE : Extract2   : Pixel @ 0 x 8                             : Source Red = 197 : Data Red = 198 : Positive Binary = '1'
VERBOSE : Extract1   : Pixel @ 0 x 8                             : Source RGB = 197    86   120 : Data RGB = 198    86   120 : Positive Binary = '1'   'X'   'X'
VERBOSE : Extract1   : Pixel @ 0 x 9                             : Source RGB = 243     4    75 : Data RGB = 243     4    75 : Positive Binary = 'X'   'X'   'X'
VERBOSE : Extract2   : Pixel @ 0 x 10                            : Source Green = 140 : Data Green = 140 : Positive Binary = '0'
VERBOSE : Extract2   : Pixel @ 0 x 10                            : Source Blue = 147 : Data Blue = 147 : Positive Binary = '0'
VERBOSE : Extract1   : Pixel @ 0 x 10                            : Source RGB = 106   140   147 : Data RGB = 106   140   147 : Positive Binary = 'X'   '0'   '0'
```

<-- CUT -->

```
STATUS  : Extract    : Processed Percent                         : 99.999999999999999%
STATUS  : Extract    : Processed Pixels                          : 2500
STATUS  : Extract    : Elapsed Time                              : 1587 Milliseconds
STATUS  : Extract    :
STATUS  : Extract    : Total Binary Characters Found             : 3191
VERBOSE : Extract2   :                                           :
VERBOSE : Extract2   : Generating Extract2 Logs                  : /Users/foocrypt/Extract.log
VERBOSE : Extract2   :                                           :
VERBOSE : Extract2   : Extracted Binary Data                     : 000111000
VERBOSE : Extract2   : Extracted Binary Data                     : 001111001
VERBOSE : Extract2   : Extracted Binary Data                     : 000111001
VERBOSE : Extract2   : Extracted Binary Data                     : 001101101
VERBOSE : Extract2   : Extracted Binary Data                     : 000111000
```

- **FooStegCypher**

  FooStegCypher adds an extra layer of security by enhancing the Brute Strength of the data images when they are stored / transmitted via the implementation of Cypher based routines by reordering the ScanMap into a repeatable CypherMap based on a user supplied KEY that is factored in **ROUNDS** via a FooSteg generated TOKEN.

FooStegCypher performs the following routines :

- **FooSteg -a [Write | Test | TestVerbose] -k**
  - User is prompted to enter [FooStegKey]
  - FooSteg Generates [FooStegToken]
  - FooSteg Accepts User Supplied [FooStegToken] -a [Write] -t | -T [FooStegToken]
  - ScanMap -> CypherMap -> [WriteMap | ExtractMap]

- **FooSteg -a Extract -k -t**
  - User is prompted to enter [FooStegKey]
  - User is prompted to enter [FooStegToken]
  - ScanMap -> CypherMap -> [ExtractMap]

- # FooSteg Verbose Switches
  - FooSteg -v [Analyse | B64Data | Copy | CypherMap | Extract1 | Extract2 | ExtractMap | Random | Read | ReadData | ScanMap | Verify1 | Verify1E | Verify2 | Verify2E | Verify3 | Verify3E | Write1 | Write2 | WriteMap | Test]
    - `Requires -l To Save Verbose Processing Logs To LogFile`
    - `Requires -V To Send Verbose Processing Logs To StdOut`
    - `Optional -i To Save Each Verbose Switch To A Seperate LogFile`

    FooSteg Verbose settings can be aggregated by separating each verbose setting by a comma.

    ie: -v Analyse,Extract1,Verify,Extract2

    Or by using the '-v Test' option to select all verbose settings.

    ie: -v Test

- # Example Verbose StdOut / LogFile Contents
  - ## Analyse

```
VERBOSE : Analyse  :                              :
VERBOSE : Analyse  : Generating Analyse Logs      : /Users/foocrypt/Analyse.log
VERBOSE : Analyse  :                              :
VERBOSE : Analyse  :
VERBOSE : Analyse  : Pixels Counted Red           : 492161
VERBOSE : Analyse  : Pixels Counted Green         : 492212
VERBOSE : Analyse  : Pixels Counted Blue          : 492161
VERBOSE : Analyse  :
VERBOSE : Analyse  :
VERBOSE : Analyse  : Pixels Counted Red           : 984430
VERBOSE : Analyse  : Pixels Counted Green         : 984348
VERBOSE : Analyse  : Pixels Counted Blue          : 984371
VERBOSE : Analyse  :
```

  - ## B64Data

```
VERBOSE : B64Data  : Generating B64Data Logs      : /Users/foocrypt/Test.log
VERBOSE : B64Data  :
VERBOSE : B64Data  : Output File Name Data Start :
VERBOSE : B64Data  : Line 1                      : iVBORw0KGgoAAAANSUhEUgAAAAoAAAAKCAYAAAH6Nf8rAAABqElEQVQYlQXBW0hT
VERBOSE : B64Data  : Line 2                      : AQAG4P+MoROWIeEepGjMy5rgrYyg6dhAXVGksOmD4ZOKY04YmQV7StEKGcNLUQli
VERBOSE : B64Data  : Line 3                      : JjFEKq8MQbRMYYoXmNKGMJiCisrcmkM5Xfz9PjQVfGJubSrR4syl6/cR4ZscJTRG
VERBOSE : B64Data  : Line 4                      : ExHdNrDlXgK17o8UqpsOaFcNQaIvE+HbAaSrtj/YUnoA1eocx6s6WeF9ycvUMLHP
VERBOSE : B64Data  : Line 5                      : SamnNAJLfBYZ3cmI6OToXUkDzNla/njRzfTbMVY9URC2n5TgURemt17BP/cG18S3
VERBOSE : B64Data  : Line 6                      : yFQPQyp7L8GHMQMiD01YF4x4ra+D0GB4zlsy4l0wDKNiD98KyiHJW4pDnRyDtWcR
VERBOSE : B64Data  : Line 7                      : gYQgNmquQFgTLWz3+XH8qwi478Hdwkpc9c4j/U4ZjNctaFsW0ePvBQ4CtXymvkT3
VERBOSE : B64Data  : Line 8                      : jT4m7Uf4eeUfZ1L+csZewS+NUwzMD/F/2ED07+Qwbj1mlzeTC06RmolinpebOXoS
VERBOSE : B64Data  : Line 9                      : ZfUDGRXupxwsdlDYF2/y9HAattavKMnOR8PJJPYGZHAM6hD1haDbPkMsKwwo9Wq6
VERBOSE : B64Data  : Line 10                     : RnapDW1Sbm5lc5KcHcpzqlxxmurtTFNmUfj+mBeKLM03vZyXoQAAAABJRU5ErkJg
VERBOSE : B64Data  : Line 11                     : gg==
VERBOSE : B64Data  : Output File Name Data End   :
```

  - ## Copy

```
VERBOSE : Copy     :                             :
VERBOSE : Copy     : Generating Copy Logs        : /Users/foocrypt/Copy.log
VERBOSE : Copy     :                             :
VERBOSE : Copy     : Pixel @ 4999 x 999          : RGB =  72   171    87 : ADD = 'X' 'X' 'X' : Write =  72   171    87
VERBOSE : Copy     : Pixel @ 4998 x 999          : RGB = 207   196   175 : ADD = 'X' 'X' 'X' : Write = 207   196   175
VERBOSE : Copy     : Pixel @ 4997 x 999          : RGB =  44    33   252 : ADD = 'X' 'X' 'X' : Write =  44    33   252
VERBOSE : Copy     : Pixel @ 4996 x 999          : RGB =  89   252   158 : ADD = 'X' 'X' 'X' : Write =  89   252   158
VERBOSE : Copy     : Pixel @ 4995 x 999          : RGB = 162    51   218 : ADD = 'X' 'X' 'X' : Write = 162    51   218
VERBOSE : Copy     : Pixel @ 4994 x 999          : RGB = 239   211    79 : ADD = 'X' 'X' 'X' : Write = 239   211    79
VERBOSE : Copy     : Pixel @ 4993 x 999          : RGB = 146    17    86 : ADD = 'X' 'X' 'X' : Write = 146    17    86
VERBOSE : Copy     : Pixel @ 4992 x 999          : RGB = 192   163   251 : ADD = 'X' 'X' 'X' : Write = 192   163   251
VERBOSE : Copy     : Pixel @ 4991 x 999          : RGB =   5   173    67 : ADD = 'X' 'X' 'X' : Write =   5   173    67
VERBOSE : Copy     : Pixel @ 4990 x 999          : RGB = 108   244   227 : ADD = 'X' 'X' 'X' : Write = 108   244   227
VERBOSE : Copy     : Pixel @ 4989 x 999          : RGB =  72   109   191 : ADD = 'X' 'X' 'X' : Write =  72   109   191
VERBOSE : Copy     : Pixel @ 4988 x 999          : RGB =  50   131   141 : ADD = 'X' 'X' 'X' : Write =  50   131   141
VERBOSE : Copy     : Pixel @ 4987 x 999          : RGB =  49   128   226 : ADD = 'X' 'X' 'X' : Write =  49   128   226
VERBOSE : Copy     : Pixel @ 4986 x 999          : RGB = 104    54    33 : ADD = 'X' 'X' 'X' : Write = 104    54    33
VERBOSE : Copy     : Pixel @ 4985 x 999          : RGB =  65   165    62 : ADD = 'X' 'X' 'X' : Write =  65   165    62
VERBOSE : Copy     : Pixel @ 4984 x 999          : RGB =  42    30   221 : ADD = 'X' 'X' 'X' : Write =  42    30   221
VERBOSE : Copy     : Pixel @ 4983 x 999          : RGB = 196    19    85 : ADD = 'X' 'X' 'X' : Write = 196    19    85
VERBOSE : Copy     : Pixel @ 4982 x 999          : RGB =  13   247    55 : ADD = 'X' 'X' 'X' : Write =  13   247    55
VERBOSE : Copy     : Pixel @ 4981 x 999          : RGB = 127   194   233 : ADD = 'X' 'X' 'X' : Write = 127   194   233
VERBOSE : Copy     : Pixel @ 4980 x 999          : RGB =  53    58    10 : ADD = 'X' 'X' 'X' : Write =  53    58    10
```

• CypherMap

```
VERBOSE : CypherMap  :
VERBOSE : CypherMap  : Generating CypherMap Logs                : /Users/foocrypt/Write.log
VERBOSE : CypherMap  :
VERBOSE : CypherMap  :
VERBOSE : CypherMap  : Creating Verbose Cypher Map              :
VERBOSE : CypherMap  : Image Width                              : 1000
VERBOSE : CypherMap  : Image Height                             : 1000
VERBOSE : CypherMap  : Total Pixels                             : 1000000
VERBOSE : CypherMap  :
VERBOSE : CypherMap  : Processed Percent                        : 0%
VERBOSE : CypherMap  : Processed Pixels                         : 0
VERBOSE : CypherMap  : Elapsed Time                             : 3559 Milliseconds
VERBOSE : CypherMap  :                                          : Pixel 0 @ 10x0
VERBOSE : CypherMap  :                                          : Pixel 1 @ 100x35
VERBOSE : CypherMap  :                                          : Pixel 2 @ 100x59
VERBOSE : CypherMap  :                                          : Pixel 3 @ 100x81
VERBOSE : CypherMap  :                                          : Pixel 4 @ 100x85
VERBOSE : CypherMap  :                                          : Pixel 5 @ 100x95
VERBOSE : CypherMap  :                                          : Pixel 6 @ 100x96
VERBOSE : CypherMap  :                                          : Pixel 7 @ 100x109
VERBOSE : CypherMap  :                                          : Pixel 8 @ 100x131
VERBOSE : CypherMap  :                                          : Pixel 9 @ 100x138
VERBOSE : CypherMap  :                                          : Pixel 10 @ 100x209
VERBOSE : CypherMap  :                                          : Pixel 11 @ 100x210
VERBOSE : CypherMap  :                                          : Pixel 12 @ 100x215
```

• Extract1

```
VERBOSE : Extract1 :                          :
VERBOSE : Extract1 : Generating Extract1 Logs : /Users/foocrypt/Extract.log
VERBOSE : Extract1 :                          :
VERBOSE : Extract1 : Pixel @ 4999 x 999       : Read RGB =  72   171    87 : Write RGB =  72   171    88 : Binary = '0'   '0'   '1'
VERBOSE : Extract1 : Pixel @ 4998 x 999       : Read RGB = 207   196   175 : Write RGB = 207   196   176 : Binary = '0'   '0'   '1'
VERBOSE : Extract1 : Pixel @ 4997 x 999       : Read RGB =  44    33   252 : Write RGB =  44    33   252 : Binary = '0'   '0'   '0'
VERBOSE : Extract1 : Pixel @ 4996 x 999       : Read RGB =  89   252   158 : Write RGB =  89   252   158 : Binary = '0'   '0'   '0'
VERBOSE : Extract1 : Pixel @ 4995 x 999       : Read RGB = 162    51   218 : Write RGB = 163    52   218 : Binary = '1'   '1'   '0'
VERBOSE : Extract1 : Pixel @ 4994 x 999       : Read RGB = 239   211    79 : Write RGB = 240   211    79 : Binary = '1'   '0'   '0'
VERBOSE : Extract1 : Pixel @ 4993 x 999       : Read RGB = 146    17    86 : Write RGB = 146    17    87 : Binary = '0'   '0'   '1'
VERBOSE : Extract1 : Pixel @ 4992 x 999       : Read RGB = 192   163   251 : Write RGB = 193   164   251 : Binary = '1'   '1'   '0'
VERBOSE : Extract1 : Pixel @ 4991 x 999       : Read RGB =   5   173    67 : Write RGB =   5   174    68 : Binary = '0'   '1'   '1'
VERBOSE : Extract1 : Pixel @ 4990 x 999       : Read RGB = 108   244   227 : Write RGB = 108   244   228 : Binary = '0'   '0'   '1'
VERBOSE : Extract1 : Pixel @ 4989 x 999       : Read RGB =  72   109   191 : Write RGB =  72   109   192 : Binary = '0'   '0'   '1'
VERBOSE : Extract1 : Pixel @ 4988 x 999       : Read RGB =  50   131   141 : Write RGB =  50   131   142 : Binary = '0'   '0'   '1'
VERBOSE : Extract1 : Pixel @ 4987 x 999       : Read RGB =  49   128   226 : Write RGB =  49   128   227 : Binary = '0'   '0'   '1'
VERBOSE : Extract1 : Pixel @ 4986 x 999       : Read RGB = 104    54    33 : Write RGB = 104    54    33 : Binary = '0'   '0'   '0'
VERBOSE : Extract1 : Pixel @ 4985 x 999       : Read RGB =  65   165    62 : Write RGB =  65   165    63 : Binary = '0'   '0'   '1'
VERBOSE : Extract1 : Pixel @ 4984 x 999       : Read RGB =  42    30   221 : Write RGB =  42    30   222 : Binary = '0'   '0'   '1'
VERBOSE : Extract1 : Pixel @ 4983 x 999       : Read RGB = 196    19    85 : Write RGB = 196    19    86 : Binary = '0'   '0'   '1'
VERBOSE : Extract1 : Pixel @ 4982 x 999       : Read RGB =  13   247    55 : Write RGB =  13   247    55 : Binary = '0'   '0'   '0'
VERBOSE : Extract1 : Pixel @ 4981 x 999       : Read RGB = 127   194   233 : Write RGB = 127   194   233 : Binary = '0'   '0'   '0'
VERBOSE : Extract1 : Pixel @ 4980 x 999       : Read RGB =  53    58    10 : Write RGB =  54    59    10 : Binary = '1'   '1'   '0'
```

• Extract2

```
VERBOSE : Extract2 :                            :
VERBOSE : Extract2 : Generating Extract2 Logs   : /Users/foocrypt/Extract.log
VERBOSE : Extract2 :                            :
VERBOSE : Extract2   : Pixel @ 0 x 0                    : Source Red = 147 : Data Red = 147 : Positive Binary = '0'
VERBOSE : Extract2   : Pixel @ 0 x 0                    : Source Blue = 124 : Data Blue = 124 : Positive Binary = '0'
VERBOSE : Extract2   : Pixel @ 0 x 1                    : Source Green = 165 : Data Green = 165 : Positive Binary = '0'
VERBOSE : Extract2   : Pixel @ 0 x 2                    : Source Red = 150 : Data Red = 151 : Positive Binary = '1'
VERBOSE : Extract2   : Pixel @ 0 x 2                    : Source Green = 230 : Data Green = 231 : Positive Binary = '1'
VERBOSE : Extract2   : Pixel @ 0 x 2                    : Source Blue = 184 : Data Blue = 185 : Positive Binary = '1'
VERBOSE : Extract2   : Pixel @ 0 x 3                    : Source Green = 218 : Data Green = 218 : Positive Binary = '0'
VERBOSE : Extract2   : Pixel @ 0 x 4                    : Source Red = 227 : Data Red = 227 : Positive Binary = '0'
VERBOSE : Extract2   : Pixel @ 0 x 4                    : Source Green = 224 : Data Green = 224 : Positive Binary = '0'
VERBOSE : Extract2   : Pixel @ 0 x 4                    : Source Blue = 160 : Data Blue = 160 : Positive Binary = '0'
VERBOSE : Extract2   : Pixel @ 0 x 5                    : Source Red = 224 : Data Red = 224 : Positive Binary = '0'
VERBOSE : Extract2   : Pixel @ 0 x 5                    : Source Green = 224 : Data Green = 225 : Positive Binary = '1'
VERBOSE : Extract2   : Pixel @ 0 x 5                    : Source Blue = 186 : Data Blue = 187 : Positive Binary = '1'
VERBOSE : Extract2   : Pixel @ 0 x 7                    : Source Blue = 138 : Data Blue = 139 : Positive Binary = '1'
VERBOSE : Extract2   : Pixel @ 0 x 8                    : Source Red = 197 : Data Red = 198 : Positive Binary = '1'
VERBOSE : Extract2   : Pixel @ 0 x 10                   : Source Green = 140 : Data Green = 140 : Positive Binary = '0'
VERBOSE : Extract2   : Pixel @ 0 x 10                   : Source Blue = 147 : Data Blue = 147 : Positive Binary = ‘0'
```

<— CUT —>

```
VERBOSE : Extract2   : Extracted Binary Data            : 000111000
VERBOSE : Extract2   : Extracted Binary Data            : 001111001
VERBOSE : Extract2   : Extracted Binary Data            : 000111001
VERBOSE : Extract2   : Extracted Binary Data            : 001101101
VERBOSE : Extract2   : Extracted Binary Data            : 000111000
VERBOSE : Extract2   : Extracted Binary Data            : 001010010
VERBOSE : Extract2   : Extracted Binary Data            : 001101100
VERBOSE : Extract2   : Extracted Binary Data            : 001111010
VERBOSE : Extract2   : Extracted Binary Data            : 001011010
```

• ExtractMap

```
VERBOSE : ExtractMap :                                  :
VERBOSE : ExtractMap : Generating ExtractMap Logs       : /Users/foocrypt/Extract.log
VERBOSE : ExtractMap :                                  :
VERBOSE : ExtractMap :                                  : Pixel 0 @ 10x0
VERBOSE : ExtractMap :                                  : Pixel 1 @ 100x35
VERBOSE : ExtractMap :                                  : Pixel 2 @ 100x59
VERBOSE : ExtractMap :                                  : Pixel 3 @ 100x81
VERBOSE : ExtractMap :                                  : Pixel 4 @ 100x85
VERBOSE : ExtractMap :                                  : Pixel 5 @ 100x95
VERBOSE : ExtractMap :                                  : Pixel 6 @ 100x96
VERBOSE : ExtractMap :                                  : Pixel 7 @ 100x109
VERBOSE : ExtractMap :                                  : Pixel 8 @ 100x131
VERBOSE : ExtractMap :                                  : Pixel 9 @ 100x138
VERBOSE : ExtractMap :                                  : Pixel 10 @ 100x209
VERBOSE : ExtractMap :                                  : Pixel 11 @ 100x210
VERBOSE : ExtractMap :                                  : Pixel 12 @ 100x215
VERBOSE : ExtractMap :                                  : Pixel 13 @ 100x222
VERBOSE : ExtractMap :                                  : Pixel 14 @ 100x230
VERBOSE : ExtractMap :                                  : Pixel 15 @ 100x242
```

• Random

```
VERBOSE :
VERBOSE : Random     :                                  :
VERBOSE : Random     : Generating Random Logs           : /Users/foocrypt/Random.log
VERBOSE : Random     :                                  :
VERBOSE :
VERBOSE : Random     : Pixel @ 0 x 0                    : RGB =  64   243   183
VERBOSE : Random     : Pixel @ 0 x 1                    : RGB = 179    70   225
VERBOSE : Random     : Pixel @ 0 x 2                    : RGB =  90    78     6
VERBOSE : Random     : Pixel @ 0 x 3                    : RGB = 163   232    16
VERBOSE : Random     : Pixel @ 0 x 4                    : RGB =  15   239     2
VERBOSE : Random     : Pixel @ 0 x 5                    : RGB = 108    14   139
VERBOSE : Random     : Pixel @ 0 x 6                    : RGB =  23    37    50
VERBOSE : Random     : Pixel @ 0 x 7                    : RGB = 234   145    27
VERBOSE : Random     : Pixel @ 0 x 8                    : RGB =  64   180    79
VERBOSE : Random     : Pixel @ 0 x 9                    : RGB =  92   182   148
VERBOSE : Random     : Pixel @ 0 x 10                   : RGB = 104    64   104
```

• Read

```
VERBOSE :
VERBOSE : Read       :
VERBOSE : Read       : Pixel Finger Print               : Starting
VERBOSE : Read       :
VERBOSE : Read       : Generating Read Logs             : /Users/foocrypt/Read.log
VERBOSE : Read       :
VERBOSE :
VERBOSE : Read       : Finger Print                     : Pixel @ 0 x 0 : RGB =  64   243   183
VERBOSE : Read       : Finger Print                     : Pixel @ 0 x 1 : RGB = 179    70   225
VERBOSE : Read       : Finger Print                     : Pixel @ 0 x 2 : RGB =  90    78     6
VERBOSE : Read       : Finger Print                     : Pixel @ 0 x 3 : RGB = 163   232    16
VERBOSE : Read       : Finger Print                     : Pixel @ 0 x 4 : RGB =  15   239     2
VERBOSE : Read       : Finger Print                     : Pixel @ 0 x 5 : RGB = 108    14   139
VERBOSE : Read       : Finger Print                     : Pixel @ 0 x 6 : RGB =  23    37    50
VERBOSE : Read       : Finger Print                     : Pixel @ 0 x 7 : RGB = 234   145    27
VERBOSE : Read       : Finger Print                     : Pixel @ 0 x 8 : RGB =  64   180    79
VERBOSE : Read       : Finger Print                     : Pixel @ 0 x 9 : RGB =  92   182   148
VERBOSE : Read       : Finger Print                     : Pixel @ 0 x 10 : RGB = 104    64   104
```

• ReadData

```
VERBOSE : ReadData :                                :
VERBOSE : ReadData : Generating ReadData Logs       : /Users/foocrypt/Extract.log
VERBOSE : ReadData :                                :
VERBOSE : ReadData :                                :
0010010000001101000011100110010010010010000010010010000001101000001101010010010110001100010001110000010
0011001000001000110101001111001000111001000110011001011010001001001001100010001010010000111001001001001
0110110000110010100100001100011100000111101000111011000101000100100111000110110011001010000110011001010
1010001001001001100010001010010000111001001001001001101100001100101001000011000111000 0
```

• ScanMap

```
VERBOSE : ScanMap    :
VERBOSE : ScanMap    : Generating ScanMap Logs                  : /Users/foocrypt/Extract.log
VERBOSE : ScanMap    :
VERBOSE : ScanMap    :                                          :
VERBOSE : ScanMap    : Processing                               : Start
VERBOSE : ScanMap    :                                          :
VERBOSE : ScanMap    :                                          : Pixel 0 @ 0x0
VERBOSE : ScanMap    :                                          : Pixel 1 @ 0x1
VERBOSE : ScanMap    :                                          : Pixel 2 @ 0x2
VERBOSE : ScanMap    :                                          : Pixel 3 @ 0x3
VERBOSE : ScanMap    :                                          : Pixel 4 @ 0x4
VERBOSE : ScanMap    :                                          : Pixel 5 @ 0x5
VERBOSE : ScanMap    :                                          : Pixel 6 @ 0x6
VERBOSE : ScanMap    :                                          : Pixel 7 @ 0x7
VERBOSE : ScanMap    :                                          : Pixel 8 @ 0x8
VERBOSE : ScanMap    :                                          : Pixel 9 @ 0x9
VERBOSE : ScanMap    :                                          : Pixel 10 @ 0x10
VERBOSE : ScanMap    :                                          : Pixel 11 @ 0x11
VERBOSE : ScanMap    :                                          : Pixel 12 @ 0x12
VERBOSE : ScanMap    :                                          : Pixel 13 @ 0x13
VERBOSE : ScanMap    :                                          : Pixel 14 @ 0x14
VERBOSE : ScanMap    :                                          : Pixel 15 @ 0x15
```

• Verify1

```
VERBOSE : Verify1  :                                :
VERBOSE : Verify1  : Generating Verify1  Logs       : /Users/foocrypt/Extract.log
VERBOSE : Verify1  :                                :
VERBOSE : Verify1  : Found Numeric                  : '001001000'
VERBOSE : Verify1  : Found Numeric                  : '72'
VERBOSE : Verify1  : Found Numeric                  : '000110100'
VERBOSE : Verify1  : Found Numeric                  : '52'
VERBOSE : Verify1  : Found Numeric                  : '001110011'
VERBOSE : Verify1  : Found Numeric                  : '115'
VERBOSE : Verify1  : Found Numeric                  : '001001001'
VERBOSE : Verify1  : Found Numeric                  : '73'
VERBOSE : Verify1  : Found Numeric                  : '001000001'
VERBOSE : Verify1  : Found Numeric                  : '65'
VERBOSE : Verify1  : Found Numeric                  : '001001000'
VERBOSE : Verify1  : Found Numeric                  : '72'
VERBOSE : Verify1  : Found Numeric                  : '000110100'
VERBOSE : Verify1  : Found Numeric                  : '52'
VERBOSE : Verify1  : Found Numeric                  : '000110101'
VERBOSE : Verify1  : Found Numeric                  : '53'
VERBOSE : Verify1  : Found Numeric                  : '001001011'
```

• Verify1E

```
VERBOSE : Verify1E :                                :
VERBOSE : Verify1E : Generating Verify1E Logs       : /Users/foocrypt/Extract.log
VERBOSE : Verify1E :                                :
VERBOSE : Verify1E : NON Numeric Characters Found   : ‘A'
VERBOSE : Verify1E : NON Numeric Characters Found   : 'z'
VERBOSE : Verify1E : NON Numeric Characters Found   : ‘#'
VERBOSE : Verify1E : NON Numeric Characters Found   : 'w'
VERBOSE : Verify1E : NON Numeric Characters Found   : 'A'
VERBOSE : Verify1E : NON Numeric Characters Found   : 'q'
VERBOSE : Verify1E : NON Numeric Characters Found   : 'g'
VERBOSE : Verify1E : NON Numeric Characters Found   : ‘*'
VERBOSE : Verify1E : NON Numeric Characters Found   : ‘)'
VERBOSE : Verify1E : NON Numeric Characters Found   : ‘@'
VERBOSE : Verify1E : NON Numeric Characters Found   : 'o'
```

• Verify2

```
VERBOSE : Verify2  :                          :
VERBOSE : Verify2  : Generating Verify2 Logs  : /Users/foocrypt/Copy.log
VERBOSE : Verify2  :                          :
VERBOSE : Verify2  : Verified                 : Pixel 0 @ 4999x999 : Write =  72 171  87 : Read =  72 171  87
VERBOSE : Verify2  : Verified                 : Pixel 1 @ 4998x999 : Write = 207 196 175 : Read = 207 196 175
VERBOSE : Verify2  : Verified                 : Pixel 2 @ 4997x999 : Write =  44  33 252 : Read =  44  33 252
VERBOSE : Verify2  : Verified                 : Pixel 3 @ 4996x999 : Write =  89 252 158 : Read =  89 252 158
VERBOSE : Verify2  : Verified                 : Pixel 4 @ 4995x999 : Write = 162  51 218 : Read = 162  51 218
VERBOSE : Verify2  : Verified                 : Pixel 5 @ 4994x999 : Write = 239 211  79 : Read = 239 211  79
VERBOSE : Verify2  : Verified                 : Pixel 6 @ 4993x999 : Write = 146  17  86 : Read = 146  17  86
VERBOSE : Verify2  : Verified                 : Pixel 7 @ 4992x999 : Write = 192 163 251 : Read = 192 163 251
VERBOSE : Verify2  : Verified                 : Pixel 8 @ 4991x999 : Write =   5 173  67 : Read =   5 173  67
VERBOSE : Verify2  : Verified                 : Pixel 9 @ 4990x999 : Write = 108 244 227 : Read = 108 244 227
VERBOSE : Verify2  : Verified                 : Pixel 10 @ 4989x999 : Write =  72 109 191 : Read =  72 109 191
VERBOSE : Verify2  : Verified                 : Pixel 11 @ 4988x999 : Write =  50 131 141 : Read =  50 131 141
VERBOSE : Verify2  : Verified                 : Pixel 12 @ 4987x999 : Write =  49 128 226 : Read =  49 128 226
```

• Verify2

```
VERBOSE : Verify2  :                          :
VERBOSE : Verify2  : Generating Verify2 Logs  : /Users/foocrypt/Write.log
VERBOSE : Verify2  :
VERBOSE : Verify2  : Numeric Character Found  : '0'
VERBOSE : Verify2  : Numeric Character Found  : '0'
VERBOSE : Verify2  : Numeric Character Found  : '1'
VERBOSE : Verify2  : Numeric Character Found  : '0'
VERBOSE : Verify2  : Numeric Character Found  : '0'
VERBOSE : Verify2  : Numeric Character Found  : '1'
VERBOSE : Verify2  : Numeric Character Found  : '0'
VERBOSE : Verify2  : Numeric Character Found  : '0'
VERBOSE : Verify2  : Numeric Character Found  : '0'
VERBOSE : Verify2  : Numeric Character Found  : '7'
VERBOSE : Verify2  : Numeric Character Found  : '2'
VERBOSE : Verify2  : Numeric Character Found  : '0'
VERBOSE : Verify2  : Numeric Character Found  : '0'
VERBOSE : Verify2  : Numeric Character Found  : '0'
VERBOSE : Verify2  : Numeric Character Found  : '1'
VERBOSE : Verify2  : Numeric Character Found  : '1'
VERBOSE : Verify2  : Numeric Character Found  : '0'
VERBOSE : Verify2  : Numeric Character Found  : '1'
VERBOSE : Verify2  : Numeric Character Found  : '0'
VERBOSE : Verify2  : Numeric Character Found  : ‘0'
```

• Verify2E

```
VERBOSE : Verify2E :
VERBOSE : Verify2E : Generating Verify2E Logs  : /Users/foocrypt/Test.log
VERBOSE : Verify2E :                           :
VERBOSE : Verify2E : FAILED                    : Pixel 0 @ 99x99 : Write =  81 208 150 : Read =  81 208 150
VERBOSE : Verify2E : FAILED                    : Pixel 1 @ 98x99 : Write =   8  72 164 : Read =   8  72 164
VERBOSE : Verify2E : FAILED                    : Pixel 2 @ 97x99 : Write =  79 184  33 : Read =  79 184  33
VERBOSE : Verify2E : FAILED                    : Pixel 3 @ 96x99 : Write =  71 116 117 : Read =  71 116 117
VERBOSE : Verify2E : FAILED                    : Pixel 4 @ 95x99 : Write = 162 254  67 : Read = 162 254  67
VERBOSE : Verify2E : FAILED                    : Pixel 5 @ 94x99 : Write = 176   7  90 : Read = 176   7  90
VERBOSE : Verify2E : FAILED                    : Pixel 6 @ 93x99 : Write =  43 138  76 : Read =  43 138  76
VERBOSE : Verify2E : FAILED                    : Pixel 7 @ 92x99 : Write =  54  65 221 : Read =  54  65 221
VERBOSE : Verify2E : FAILED                    : Pixel 8 @ 91x99 : Write = 244  82 255 : Read = 244  82 255
VERBOSE : Verify2E : FAILED                    : Pixel 9 @ 90x99 : Write = 178 194 180 : Read = 178 194 180
VERBOSE : Verify2E : FAILED                    : Pixel 10 @ 89x99 : Write = 229  86 228 : Read = 229  86 228
VERBOSE : Verify2E : FAILED                    : Pixel 11 @ 88x99 : Write = 212  85  68 : Read = 212  85  68
VERBOSE : Verify2E : FAILED                    : Pixel 12 @ 87x99 : Write =  31  93  32 : Read =  31  93  32
VERBOSE : Verify2E : FAILED                    : Pixel 13 @ 86x99 : Write =  31 239  65 : Read =  31 239  65
```

• Verify3

```
VERBOSE : Verify3    :
VERBOSE : Verify3    : Generating Verify3 Logs                  : /Users/foocrypt/Write.log
VERBOSE : Verify3    :
VERBOSE : Verify3    : Verified                                 : Pixel 0 @ 10x0 : Write = 202  97 241 : Read = 202  97 241
VERBOSE : Verify3    : Verified                                 : Pixel 1 @ 100x35 : Write = 150 124  89 : Read = 150 124  89
VERBOSE : Verify3    : Verified                                 : Pixel 2 @ 100x59 : Write = 234 127 184 : Read = 234 127 184
VERBOSE : Verify3    : Verified                                 : Pixel 3 @ 100x81 : Write = 109 237 202 : Read = 109 237 202
VERBOSE : Verify3    : Verified                                 : Pixel 4 @ 100x85 : Write =  84  55   1 : Read =  84  55   1
VERBOSE : Verify3    : Verified                                 : Pixel 5 @ 100x95 : Write =  57 114  58 : Read =  57 114  58
VERBOSE : Verify3    : Verified                                 : Pixel 6 @ 100x96 : Write = 147 183 168 : Read = 147 183 168
VERBOSE : Verify3    : Verified                                 : Pixel 7 @ 100x109 : Write = 251   2 170 : Read = 251   2 170
VERBOSE : Verify3    : Verified                                 : Pixel 8 @ 100x131 : Write =  53  54 169 : Read =  53  54 169
VERBOSE : Verify3    : Verified                                 : Pixel 9 @ 100x138 : Write =  64 232 155 : Read =  64 232 155
VERBOSE : Verify3    : Verified                                 : Pixel 10 @ 100x209 : Write =   3 242  37 : Read =   3 242  37
VERBOSE : Verify3    : Verified                                 : Pixel 11 @ 100x210 : Write = 208  55 255 : Read = 208  55 255
VERBOSE : Verify3    : Verified                                 : Pixel 12 @ 100x215 : Write = 182  29  75 : Read = 182  29  75
VERBOSE : Verify3    : Verified                                 : Pixel 13 @ 100x222 : Write = 102 231  35 : Read = 102 231  35
VERBOSE : Verify3    : Verified                                 : Pixel 14 @ 100x230 : Write =  16  87  59 : Read =  16  87  59
VERBOSE : Verify3    : Verified                                 : Pixel 15 @ 100x242 : Write = 220  35 252 : Read = 220  35 252
```

- # Verify3E

```
VERBOSE : Verify3E   :
VERBOSE : Verify3E   : Generating Verify3E Logs                    : /Users/foocrypt/Write.log
VERBOSE : Verify3E   :
VERBOSE : Verify3E   : FAILED                                      : Pixel 0 @ 10x0 : Write = 202  97 241 : Read = 202  97 240
VERBOSE : Verify3E   : FAILED                                      : Pixel 1 @ 100x35 : Write = 150 124  89 : Read = 150 124  88
VERBOSE : Verify3E   : FAILED                                      : Pixel 2 @ 100x59 : Write = 234 127 184 : Read = 234 127 183
VERBOSE : Verify3E   : FAILED                                      : Pixel 3 @ 100x81 : Write = 109 237 202 : Read = 109 237 201
VERBOSE : Verify3E   : FAILED                                      : Pixel 4 @ 100x85 : Write =  84  55   1 : Read =  84  55   0
VERBOSE : Verify3E   : FAILED                                      : Pixel 5 @ 100x95 : Write =  57 114  58 : Read =  57 114  57
VERBOSE : Verify3E   : FAILED                                      : Pixel 6 @ 100x96 : Write = 147 183 168 : Read = 147 183 167
VERBOSE : Verify3E   : FAILED                                      : Pixel 7 @ 100x109 : Write = 251   2 170 : Read = 251   2 171
VERBOSE : Verify3E   : FAILED                                      : Pixel 8 @ 100x131 : Write =  53  54 169 : Read =  53  54 160
VERBOSE : Verify3E   : FAILED                                      : Pixel 9 @ 100x138 : Write =  64 232 155 : Read =  64 232 155
VERBOSE : Verify3E   : FAILED                                      : Pixel 10 @ 100x209 : Write =   3 242  37 : Read =   3 242  39
VERBOSE : Verify3E   : FAILED                                      : Pixel 11 @ 100x210 : Write = 208  55 255 : Read = 208  55 250
VERBOSE : Verify3E   : FAILED                                      : Pixel 12 @ 100x215 : Write = 182  29  75 : Read = 182  29 770
VERBOSE : Verify3E   : FAILED                                      : Pixel 13 @ 100x222 : Write = 102 231  35 : Read = 102 231  30
VERBOSE : Verify3E   : FAILED                                      : Pixel 14 @ 100x230 : Write =  16  87  59 : Read =  16  87  51
VERBOSE : Verify3E   : FAILED                                      : Pixel 15 @ 100x242 : Write = 220  35 252 : Read = 220  35 250
```

- # Write1

```
VERBOSE : Write1     : Pixel @ 0 x 0      : Source RGB = 194   87   74 : Positive Binary = '0' 'X' 'X' : Data RGB = 194   87   74
VERBOSE : Write1     : Pixel @ 0 x 1      : Source RGB =  24    7   22 : Positive Binary = 'X' 'X' 'X' : Data RGB =  24    7   22
VERBOSE : Write1     : Pixel @ 0 x 2      : Source RGB =  30   93  245 : Positive Binary = 'X' 'X' 'X' : Data RGB =  30   93  245
VERBOSE : Write1     : Pixel @ 0 x 3      : Source RGB = 185  241  229 : Positive Binary = '0' 'X' '1' : Data RGB = 185  241  230
VERBOSE : Write1     : Pixel @ 0 x 4      : Source RGB = 253  180   69 : Positive Binary = 'X' '1' 'X' : Data RGB = 253  181   69
VERBOSE : Write1     : Pixel @ 0 x 5      : Source RGB =  80    7  202 : Positive Binary = 'X' 'X' '1' : Data RGB =  80    7  203
VERBOSE : Write1     : Pixel @ 0 x 6      : Source RGB = 127   27   77 : Positive Binary = '1' 'X' 'X' : Data RGB = 128   27   77
VERBOSE : Write1     : Pixel @ 0 x 7      : Source RGB =  21   83  161 : Positive Binary = 'X' 'X' '0' : Data RGB =  21   83  161
VERBOSE : Write1     : Pixel @ 0 x 8      : Source RGB = 239  138  174 : Positive Binary = 'X' '1' '0' : Data RGB = 239  139  174
VERBOSE : Write1     : Pixel @ 0 x 9      : Source RGB = 112  217  234 : Positive Binary = 'X' '0' 'X' : Data RGB = 112  217  234
VERBOSE : Write1     : Pixel @ 0 x 10     : Source RGB = 182   32   58 : Positive Binary = '0' 'X' 'X' : Data RGB = 182   32   58
```

- # Write2

```
VERBOSE : Write2     : Binary Character = 1    : Positive Write Binary Character '0' To Pixel @ 0 x 0 : 194  ->  194 : Red
VERBOSE : Write2     : Binary Character = 2    : Positive Write Binary Character '0' To Pixel @ 0 x 3 : 185  ->  185 : Red
VERBOSE : Write2     : Binary Character = 3    : Write Binary Character '1' To Pixel @ 0 x 3 : 229  ->  230 : Blue
VERBOSE : Write2     : Binary Character = 4    : Positive Write Binary Character '1' To Pixel @ 0 x 4 : 180  ->  181 : Green
VERBOSE : Write2     : Binary Character = 5    : Write Binary Character '1' To Pixel @ 0 x 5 : 202  ->  203 : Blue
VERBOSE : Write2     : Binary Character = 6    : Positive Write Binary Character '1' To Pixel @ 0 x 6 : 127  ->  128 : Red
VERBOSE : Write2     : Binary Character = 7    : Write Binary Character '0' To Pixel @ 0 x 7 : 161  ->  161 : Blue
VERBOSE : Write2     : Binary Character = 8    : Positive Write Binary Character '1' To Pixel @ 0 x 8 : 138  ->  139 : Green
VERBOSE : Write2     : Binary Character = 9    : Write Binary Character '0' To Pixel @ 0 x 8 : 174  ->  174 : Blue
VERBOSE : Write2     : Binary Character = 10   : Positive Write Binary Character '0' To Pixel @ 0 x 9 : 217  ->  217 : Green
VERBOSE : Write2     : Binary Character = 11   : Positive Write Binary Character '0' To Pixel @ 0 x 10 : 182  ->  182 : Red
VERBOSE : Write2     : Binary Character = 12   : Positive Write Binary Character '1' To Pixel @ 0 x 12 : 144  ->  145 : Red
VERBOSE : Write2     : Binary Character = 13   : Write Binary Character '1' To Pixel @ 0 x 12 : 161  ->  162 : Blue
VERBOSE : Write2     : Binary Character = 14   : Positive Write Binary Character '1' To Pixel @ 0 x 14 : 218  ->  219 : Red
VERBOSE : Write2     : Binary Character = 15   : Positive Write Binary Character '1' To Pixel @ 0 x 14 : 216  ->  217 : Green
VERBOSE : Write2     : Binary Character = 16   : Write Binary Character '0' To Pixel @ 0 x 14 : 213  ->  213 : Blue
```

- # WriteMap

```
VERBOSE : WriteMap   :                                             :
VERBOSE : WriteMap   : Generating WriteMap Logs                    : /Users/foocrypt/Write.log
VERBOSE : WriteMap   :                                             :
VERBOSE : WriteMap   :                                             : Pixel 0 @ 10x0
VERBOSE : WriteMap   :                                             : Pixel 1 @ 100x35
VERBOSE : WriteMap   :                                             : Pixel 2 @ 100x59
VERBOSE : WriteMap   :                                             : Pixel 3 @ 100x81
VERBOSE : WriteMap   :                                             : Pixel 4 @ 100x85
VERBOSE : WriteMap   :                                             : Pixel 5 @ 100x95
VERBOSE : WriteMap   :                                             : Pixel 6 @ 100x96
VERBOSE : WriteMap   :                                             : Pixel 7 @ 100x109
VERBOSE : WriteMap   :                                             : Pixel 8 @ 100x131
VERBOSE : WriteMap   :                                             : Pixel 9 @ 100x138
VERBOSE : WriteMap   :                                             : Pixel 10 @ 100x209
VERBOSE : WriteMap   :                                             : Pixel 11 @ 100x210
VERBOSE : WriteMap   :                                             : Pixel 12 @ 100x215
VERBOSE : WriteMap   :                                             : Pixel 13 @ 100x222
VERBOSE : WriteMap   :                                             : Pixel 14 @ 100x230
VERBOSE : WriteMap   :                                             : Pixel 15 @ 100x242
VERBOSE : WriteMap   :                                             : Pixel 16 @ 100x245
VERBOSE : WriteMap   :                                             : Pixel 17 @ 100x246
VERBOSE : WriteMap   :                                             : Pixel 18 @ 100x249
VERBOSE : WriteMap   :                                             : Pixel 19 @ 100x256
VERBOSE : WriteMap   :                                             : Pixel 20 @ 100x261
```

- # Test
  - **Generates ALL the Above Verbose Logging Switches.**

FooStegCypher Technical Details

FooStegCypher enhances the Brute Strength of the Data Image.

FooStegCypher takes the user supplied FooStegKey and translates the characters into their corresponding ASCII Code Values.

FooStegCypher then factors the ASCII Code Values, by the lowest Prime Number (0 - 500) which is incremented by 500 per character index, and in addition by the random 6 digit FooStegToken, to create a numerical string of values, [0-9] [CypherSortKey].

The flexibility with the end user being able to select the number of rounds (R) between 19 - 512, increases the length of the CypherSortKey, based on the length of the FooStegKey. ie: A longer CypherSortKey can be obtained by using a short FooStegKey by increasing the rounds (R).

Example Code

Such as :

```
FSK = FooStegKey [ 8 - 10240 ASCII Characters ]
FST = FooStegToken
CAP = Character ASCII Primed
SK  = SortKey
CSK = CypherSortKey
R   = Rounds [ 19 - 512 ]

Where FST = $(( RANDOM 6 Digit Number, 100000 >=< 999999 ))
Where Index = 1
until [ ${Index} > [ String Length FooStegKey ] ]
do

      CAP=$(( [ FSK character ${Index}, ASCII Code Value ] \
      * ( LPN [ ( ${Index} * 500 ) - 500 ] - ( ${Index} * 500 ) ] ))

      SK=$(( ${SK} + ${CAP} ))

      Increment Index by 1

done

while [ ( String Length SortKey ) < ( R ) ]
do

      SK=$(( ${SK } + ${SK } + ${FST} ))

      CSK="${CSK}${SK}"

done
```

SALTING

As you can determine from the sample code above, utilising the same FooStegKey will generate different SortKey due to the use of the RANDOM FooStegToken which applies a ‘SALTED’ result to the final SortKey.

Sample Results Using The Same FooStegKey :

Example 1

FooStegKey : fWMvvOJVXet
FooStegToken : 475316
CypherSortKey :
57833661204204824559412495941409966359619980250840008033280063598016017472763203969868640841505212817305420256350861565127064762810254177057220508401646041016850823682033749178816406754588923281351393100656270326151613125406998348262508144720125250162941934010500325931399621000651910330842001303868193284002607783918016800521561536763360104312782668672020862604065213440417252556620268808345055885565376166901165242810752333802378017221504667604803566043009335209654663686018670419356858817203734083876124923440746816775700300688149363355187591 6

Example 2

FooStegKey : fWMvvOJVXet
FooStegToken : 842737
CypherSortKey :
61507871314431127131359551054551110536472229500314467427998943283351789499407357984155171605258391432189441528644631567572901058711145810544792291629516954583267461279166543349911833309512719366661986817573332405790871466648200091129332964844559586659305318551173318619064472346637246556314693274501539999386549011507351877309803144207375461960713115175092392151050391501847843105281530036956862948367600739137267394711201478274543216792402956549094860954805913098198149279611826196404725911922365239281787919384473047856441857576894609571296798 87

Example 3

FooStegKey : fWMvvOJVXet
FooStegToken : 388353
CypherSortKey
:56964031178115923950671482896959696774319432383938903603177846041515573091833115006719623040179112461191935249227722234984593279999692253951199384896255398770180863797540750079159508188851131901641653756380328719103127606578265592552131604147151042632471295102085265330943204170531050239408341062488831816682125366015163336425112038332667285026291196533457005646591130669140116815352613382802375142352267656047891199104535312096170751209070624192729855418141248385848063836282496772084479167256499354455731133451299870895029756690259974179394303

CypherSortKey

The CypherSortKey is split 1 character at a time, applied as a prefix character to the ScanMap, which is then sorted, to reorder the ScanMap into the CypherMap, which is then utilised as the WriteMap or ExtractMap.

Sample Results Using The Same FooStegKey with different Scan Modes :

Example 1

FooStegKey : fWMvvOJVXet
FooStegToken : 475316
CypherSortKey :
57833661204204824559412495941409966359619980250840008033280063598016017472763203969868640841505212817305420256350861565127064762810254177057220508401646041016850823682033749178816406754588923281351393100656270326151613125406998348262508144720125250162941934010500325931399621000651910330842001303868193284002607783918016800521561536763360104312782668672020862604065213440417252556620268808345055885565376166901165242810752333802378017221504667604803566043009335209654663686018670419356858817203734083876124923440746816775700300688149363355187591 6

With a Scan Mode of 0, the following ScanMap is created :

VERBOSE : ScanMap : : Pixel 0 @ 0x0
VERBOSE : ScanMap : : Pixel 1 @ 0x1
VERBOSE : ScanMap : : Pixel 2 @ 0x2
VERBOSE : ScanMap : : Pixel 3 @ 0x3
VERBOSE : ScanMap : : Pixel 4 @ 0x4
VERBOSE : ScanMap : : Pixel 5 @ 0x5
VERBOSE : ScanMap : : Pixel 6 @ 0x6
VERBOSE : ScanMap : : Pixel 7 @ 0x7
VERBOSE : ScanMap : : Pixel 8 @ 0x8
VERBOSE : ScanMap : : Pixel 9 @ 0x9
VERBOSE : ScanMap : : Pixel 10 @ 0x10

Which then is sorted to create the following CypherMap

VERBOSE : CypherMap : : Pixel 0 @ 100x6
VERBOSE : CypherMap : : Pixel 1 @ 100x11
VERBOSE : CypherMap : : Pixel 2 @ 100x20
VERBOSE : CypherMap : : Pixel 3 @ 100x24
VERBOSE : CypherMap : : Pixel 4 @ 100x30
VERBOSE : CypherMap : : Pixel 5 @ 100x40
VERBOSE : CypherMap : : Pixel 6 @ 100x48
VERBOSE : CypherMap : : Pixel 7 @ 100x55
VERBOSE : CypherMap : : Pixel 8 @ 100x60
VERBOSE : CypherMap : : Pixel 9 @ 100x62
VERBOSE : CypherMap : : Pixel 10 @ 100x65

And is utilised as the WriteMap or the ExtractMap

Example 2

FooStegKey : fWMvvOJVXet
FooStegToken : 842737
CypherSortKey :
615078713144311271313595510545511105364722295003144674279989432833517894
994073579841551716052583914321894415286446315675729010587111458105447922
916295169545832674612791665433499118333095127193666619868175733324057908
714666482000911293329648445595866593053185511733186190644723466372465563
146932745015399993865490115073518773098031442073754619607131151750923921
510503915018478431052815300369568629483676007391372673947112014782745432
167924029565490948609548059130981981492796118261964047259119223652392817
879193844730478564418575768946095712967988 7

With a Scan Mode of 1, the following ScanMap is created :

VERBOSE : ScanMap : : Pixel 0 @ 0x999
VERBOSE : ScanMap : : Pixel 1 @ 0x998
VERBOSE : ScanMap : : Pixel 2 @ 0x997
VERBOSE : ScanMap : : Pixel 3 @ 0x996
VERBOSE : ScanMap : : Pixel 4 @ 0x995
VERBOSE : ScanMap : : Pixel 5 @ 0x994
VERBOSE : ScanMap : : Pixel 6 @ 0x993
VERBOSE : ScanMap : : Pixel 7 @ 0x992
VERBOSE : ScanMap : : Pixel 8 @ 0x991
VERBOSE : ScanMap : : Pixel 9 @ 0x990
VERBOSE : ScanMap : : Pixel 10 @ 0x989

Which then is sorted to create the following CypherMap

VERBOSE : CypherMap : : Pixel 0 @ 100x999
VERBOSE : CypherMap : : Pixel 1 @ 100x994
VERBOSE : CypherMap : : Pixel 2 @ 100x989
VERBOSE : CypherMap : : Pixel 3 @ 100x984
VERBOSE : CypherMap : : Pixel 4 @ 100x962
VERBOSE : CypherMap : : Pixel 5 @ 100x930
VERBOSE : CypherMap : : Pixel 6 @ 100x911
VERBOSE : CypherMap : : Pixel 7 @ 100x895
VERBOSE : CypherMap : : Pixel 8 @ 100x872
VERBOSE : CypherMap : : Pixel 9 @ 100x864
VERBOSE : CypherMap : : Pixel 10 @ 100x853

And is utilised as the WriteMap or the ExtractMap

Example 3

FooStegKey : fWMvvOJVXet
FooStegToken : 388353
CypherSortKey
:56964031178115923950671482896959696774319432383938903603177846041515573
09183311500671962304017911246119193524922772223498459327999692253951199
38489625539877018086379754075007915950818885113190164165375638032871910 3
12760657826559255213160414715104263247129510208526533094320417053105023 9
40834106248883181668212536601516333642511203833266728502629119653345700 5
64659113066914011681535261338280237514235226765604789119910453531209617 0
75120907062419272985541814124838584806383628249677208447916725649935445 5
73113345129987089502975669025997417939430 3

With a Scan Mode of 2, the following ScanMap is created :

VERBOSE : ScanMap : : Pixel 0 @ 999x0
VERBOSE : ScanMap : : Pixel 1 @ 999x1
VERBOSE : ScanMap : : Pixel 2 @ 999x2
VERBOSE : ScanMap : : Pixel 3 @ 999x3
VERBOSE : ScanMap : : Pixel 4 @ 999x4
VERBOSE : ScanMap : : Pixel 5 @ 999x5
VERBOSE : ScanMap : : Pixel 6 @ 999x6
VERBOSE : ScanMap : : Pixel 7 @ 999x7
VERBOSE : ScanMap : : Pixel 8 @ 999x8
VERBOSE : ScanMap : : Pixel 9 @ 999x9
VERBOSE : ScanMap : : Pixel 10 @ 999x10

Which then is sorted to create the following CypherMap

VERBOSE : CypherMap : : Pixel 0 @ 899x3
VERBOSE : CypherMap : : Pixel 1 @ 899x8
VERBOSE : CypherMap : : Pixel 2 @ 899x12
VERBOSE : CypherMap : : Pixel 3 @ 899x16
VERBOSE : CypherMap : : Pixel 4 @ 899x18
VERBOSE : CypherMap : : Pixel 5 @ 989x2
VERBOSE : CypherMap : : Pixel 6 @ 899x23
VERBOSE : CypherMap : : Pixel 7 @ 899x28
VERBOSE : CypherMap : : Pixel 8 @ 899x49
VERBOSE : CypherMap : : Pixel 9 @ 899x64
VERBOSE : CypherMap : : Pixel 10 @ 899x76

And is utilised as the WriteMap or the ExtractMap

WriteMap Use

The BASE64 Data is converted into Binary Data.

The Binary Data is Written to the Data Image, as per the sequence defined in the WriteMap, taking into account the RGB Min / Max values, utilising the Source Image to determine the Binary Difference to create the Data Image RGB value.

ExtractMap Use

The Binary Data is Extracted from the Data Image, as per the sequence defined in the ExtractMap, taking into account the RGB Min / Max values, utilising the Source Image to determine the Binary Difference until 9 zeros are read from the Data Image.

The Binary Data is converted into BASE64 Data.

Acronyms and Abbreviations

TBD	To Be Disclosed
	If you discover a term you are unable to find the answer for within this documentation or online at FooCrypt.XYZ, send a request via the webform at the bottom of FooCrypt.XYZ selecting one of the “Question” options for further information.

Addendum

- **TBD**