



Version 1.1
User Manual

Introduction

An introduction to RubyEncoder

by RubyEncoder Team

This RubyEncoder User Manual covers all of the features in this new exciting product. We hope that you enjoy using our product and find this user guide to be informative.

If there is anything that you feel has been omitted from this user manual, then please let us know as we are passionate about providing excellent service.

Have fun using your new product...

RubyEncoder 1.1

Copyright 2008 rubyencoder.com

All rights reserved. No parts of this work may be reproduced in any form or by any means - graphic, electronic, or mechanical, including photocopying, recording, taping, or information storage and retrieval systems - without the written permission of the publisher.

Products that are referred to in this document may be either trademarks and/or registered trademarks of the respective owners. The publisher and the author make no claim to these trademarks.

While every precaution has been taken in the preparation of this document, the publisher and the author assume no responsibility for errors or omissions, or for damages resulting from the use of information contained in this document or from the use of programs and source code that may accompany it. In no event shall the publisher and the author be liable for any loss of profit or any other commercial damage caused or alleged to have been caused directly or indirectly by this document.

Table of Contents

Foreword	0
Part I Introduction	2
1 About RubyEncoder™	2
2 How to buy	2
3 Features	2
Part II Using of RubyEncoder™	6
1 Ultimate Ruby and Ruby on Rails Scripts Protection	6
2 Supported Ruby versions	6
3 Cross platform encryption	6
4 Command line encoder installation	6
Command line encoder installation for Linux	7
Command line encoder installation for FreeBSD	8
Command line encoder installation for Mac OS X	10
Command line encoder installation for Windows	11
5 Running the command line encoder	11
First run	11
Usage	12
Script locking options (full version only)	13
Advanced options	18
Other options	21
6 Using external script license generator (full version only)	24
Usage	25
Script locking options	25
Advanced options	29
Other options	30
7 Using information tool (full version only)	30
Usage	30
8 Ruby shell scripts encoding	31
Part III Protected script loaders	33
1 Installing loaders	33
2 rgloader directory structure	33
Part IV Errors, common mistakes and possible solutions	36
1 Encoded scripts modification	36
2 Error messages during encoding	36
3 Error messages during protected scripts run	37

Index

41

RubyEncoder 1.1

Part



1 Introduction

1.1 About RubyEncoder™

RubyEncoder has been built as professional system for source code protection. Our team of programmers, some of whom programmed and developed the successful SourceGuardian PHP Encoder (www.sourceguardian.com) have created proprietary methods for encrypting code whilst keeping the maximum flexibility for the distribution of your scripts.

We, as a team, are very excited about the potential for our product as well as the Ruby and Ruby on Rails community. We finally have a method to protect and distribute our commercial code and many of the developers who have worked with us during the beta phase have said that RubyEncoder solves many of the problems they previously had.

As for the future of RubyEncoder, we thank everyone who has purchased, downloaded or even taken the time to browse our site. We plan to continue to increase the functionality and power of these programs whilst keeping an affordable upgrade path. We always welcome new suggestions and if you feel that you have something you would like us to add, then feel free to contact us

Thanks for your interest, and thanks for your business.

The RubyEncoder Team

1.2 How to buy

To purchase RubyEncoder 1.1, please visit the following:

Website: <http://www.rubyencoder.com/purchase/index.html>

There are two methods available: via Credit/Debit Card or via Paypal.

1.3 Features

RubyEncoder 1.1 Features List

Protection method

The RubyEncoder 1.1 protects Ruby scripts by compiling Ruby source code into a bytecode format and this is followed by encryption. This protects your scripts from reverse engineering. Ruby scripts protected with RubyEncoder can be executed but cannot be used to extract Ruby source code as there is no source code remaining within the protected script in any form. Scripts protected with RubyEncoder will require installation of the RubyEncoder Loader in order to run. The RubyEncoder Loader is a compiled Ruby module which is automatically loaded and used to run the protected scripts. The RubyEncoder Loaders are binary files that are different for each OS and platform. See the [protected scripts loaders](#) section below to know more about the RubyEncoder Loaders.

Protected code

This sample Ruby code:

```
puts "Hello World!"
```

Will become like this when encoded with RubyEncoder 1.1:

```
# RubyEncoder v1.0

_d = _d0 = File.expand_path(File.dirname(__FILE__)); while 1 do _f = _d +
'/rgloader/loader.rb'; bre
ak if File.exist?(_f); _d1 = File.dirname(_d); if _d1 == _d then raise "Ruby
script '"+__FILE__+"' i
s protected by RubyEncoder and requires the RubyEncoder loader. Please visit the
http://www.rubyenco
der.com/loaders/ RubyEncoder site to download the required loader and unpack it
into '"+_d0+"/rgload
er/' directory to run this protected script."; break; else _d = _d1; end; end;
require _f; RGLoader:
:load('AAEAAAEaAAAAIAAAAA/4B5q/17q0E9u+i30t9BI31n8T3mAjkfJJgZIWWw35emHS4pdu70qVg
ASS159/rPoG8FfRRGM
sZCz2RlJno4TGXP3JZsnHBsftX/NeKgoTYNYLUYCxvBaQAHYcx6avkVQxnmsgH/qXvmEgAAAGAAAAB75w3
qzOZQCmvQDuOs+G9ZV
hz0GbAyloT4injTT83Iijyj0iubOUfKrn2+fRb7Q0m3dEXGqgUtKkGzz2yaCE0T9yV1V0ZtZv4RKezGy1U
JdYtDb4lrK0t8S9FRL
BYnwAYAAAAA');
```

Supported Ruby versions

RubyEncoder 1.1 works with the following versions: Ruby 1.8.6, 1.8.7 and 1.9.0 are fully supported.

Interface

A powerful cross-platform command line encoder is available that runs under Mac OSX, Linux, FreeBSD and Windows.

Locking

To protect your scripts from unauthorised usage RubyEncoder 1.1 has added some features that can optionally lock your scripts to run only from predefined IP addresses, domain names or LAN hardware addresses (MAC). RubyEncoder 1.1 can also easily produce trial versions of your scripts by setting an expiry date for the script or by limiting the number of days that the protected script will work. With RubyEncoder 1.1 you may optionally lock your scripts so that they require a special license file in order to run. This file may be distributed with the script or separately from it and this option gives you an opportunity to encode your script once and distribute to users with different licenses. Each license may have different and specific attributes. The RubyEncoder license generator tool may be used to create a license file directly on your server. This lets you easily provide your customers with a license file for your product at the precise moment of purchase, or downloading of a trial version, of your product.

Here is a sample list of features:

- locking to a specific date
- locking to multiple domain names (when run in a web server environment)
- locking to multiple ip addresses (when run in a web server environment)
- locking to multiple LAN hardware (MAC) addresses
- improved locking to a specific domain name with encryption (when run in a web server environment). The domain name is used as a part of the key for encryption, so protected scripts may not be decrypted and run from another domain.
- improved locking to an ip address with encryption (when run in a web server environment). The ip address is used as a part of the key for encryption. This means that the protected scripts cannot be decrypted and run from another ip address.
- the ability to lock and protect using an external license file produced by the RubyEncoder 1.1 license generator. This is ideal for creating protected scripts to be distributed between different users and it

will even allow different options for different users. The RubyEncoder 1.1 license generator tool can run under Mac OSX, Linux and Linux as a command line tool which adds another powerful element - It provides a method for licenses to be dynamically generated and this is useful (for example) when selling scripts online.

Other options

The following is not an exhaustive list, but covers some of the other options in this new version:

- option to check the date with online time servers (useful when locking to date is used)
- option to assign a custom Ruby code to act as an error handler which will catch errors related to protected script loading or locking
- option to set a custom defined constant which may be read later from the protected Ruby code
- automatic backup of source files
- multiple files processing: enumerated, file mask optionally with directory recursion or file list from the command line
- option to exclude some files from encoding process: enumerated or file masks
- option to specify an output directory for protected scripts
- encoding confirmation when run from the command line
- option to include Ruby code to run before a protected script. This is best for including copyright information or for any other advanced needs
- option to replace the standard error handler when the appropriate loader is not found. Ruby code can be included here

Cross platform

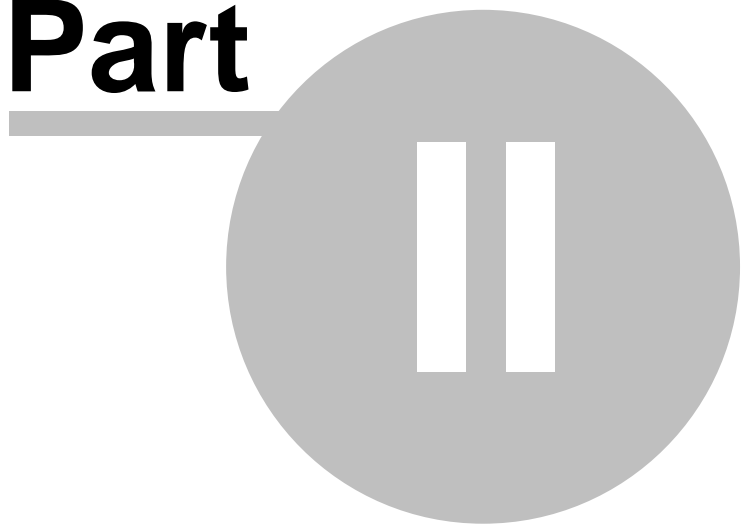
Cross platform encoding. A script encoded under one operating system will run under any other supported operating systems. Currently we have an encoder for Mac OSX, Linux and FreeBSD. Script Loaders will run under Mac OSX, Linux, FreeBSD and Windows. Also we have plans to support more operating systems for protected scripts execution such as NetBSD, OpenBSD, Solaris and others.

Evaluation

We provide a Free 7 days evaluation of RubyEncoder 1.1

RubyEncoder 1.1

Part



2 Using of RubyEncoder™

2.1 Ultimate Ruby and Ruby on Rails Scripts Protection

RubyEncoder 1.1 protects Ruby scripts by compiling Ruby source code into a bytecode format and this is followed by encryption. This protects your scripts from reverse engineering.

To protect your scripts from unauthorised usage RubyEncoder 1.1 has added features that can optionally lock your scripts to run only from predefined IP addresses, domain names or LAN hardware addresses (MAC). RubyEncoder 1.1 can also easily produce trial versions of your scripts by setting an expiry date for the script or by limiting the number of days that protected script will work. With RubyEncoder 1.1 you may optionally lock your scripts so that they require a special license file in order to run. This file may be distributed with the script or separately from it and this option gives you an opportunity to encode your scripts once and distribute to users with different licenses. Each license may have different and specific attributes. The RubyEncoder license generator tool may be used to create a license file directly on your server. This lets you easily provide your customers with a license file for your product in the moment of purchase or downloading of a trial version of your product.

2.2 Supported Ruby versions

RubyEncoder 1.1 supports encoding for Ruby 1.8.6, 1.8.7 and 1.9.0 (which is still experimental at the moment of writing this manual). There is one command line encoder which may encode your scripts for Ruby 1.8.x and/or Ruby 1.9.x. When encoding for multiple Ruby versions your Ruby scripts should be compatible with Ruby 1.8.x and 1.9.x. RubyEncoder will write separate bytecodes into the protected script for each version of Ruby. The RubyEncoder Loader will extract and run the required version of bytecode from the protected script during the protected script execution.

2.3 Cross platform encryption

RubyEncoder 1.1 protected scripts have the same internal format for all supported operating systems. This mean that scripts encoded with RubyEncoder 1.1 under Mac OS X (for example) will also run under a Linux OS with the appropriate linux loaders. This flexibility is the same for all other supported operating systems.

2.4 Command line encoder installation

You may download RubyEncoder 1.1 installation package as .zip, .tar.gz or tar.bz2 file or .dmg file for Mac OSX. Windows version includes an installer which you may run to install RubyEncoder. An evaluation version of RubyEncoder is available as a free download from our site <http://rubyencoder.com/trial.html> . The full version of RubyEncoder is available for download from your account profile after purchase.

[Click here](#) to read more about installation for Linux.

[Click here](#) to read more about installation for FreeBSD.

[Click here](#) to read more about installation for Mac OSX.

[Click here](#) to read more about installation for Windows.

2.4.1 Command line encoder installation for Linux

You need to unpack the downloaded encoder installation file into any directory you wish. For Linux we suggest that you install into the `/usr/local` or any user home directory `/home/username`. The following instruction is expecting you use a terminal for installation. The downloaded encoder file may have the following names according to the OS and version of the encoder:

```
rubyencoder-1.1-linux.tar.gz (or .zip, or .tar.bz2)
rubyencoder-1.1-eval-linux.tar.gz (or .zip, or .tar.bz2)
```

Example (trial version):

Copy the downloaded file to the destination directory `/usr/local`. You may use either OS interface or terminal to locate and copy the downloaded file:

```
> cp /your/path/to/downloaded/rubyencoder-1.1-eval-linux.tar.gz /usr/local
```

Unpack:

```
> cd /usr/local
> tar xzf rubyencoder-1.1-eval-linux.tar.gz
```

Update permissions:

```
> cd /usr/local/rubyencoder-1.1-eval/bin
> chmod a+x rubyencoder
```

Run the encoder:

```
> ./rubyencoder
```

or if you are using zip:

Copy the downloaded file to the destination directory `/usr/local`. You may use either OS interface or terminal to locate and copy the downloaded file:

```
> cp /your/path/to/downloaded/rubyencoder-1.1-eval-linux.zip /usr/local
```

Unpack:

```
> cd /usr/local
> unzip rubyencoder-1.1-eval-linux.zip
```

Update permissions:

```
> cd /usr/local/rubyencoder-1.1-eval/bin
> chmod a+x rubyencoder
```

Run the encoder:

```
> ./rubyencoder
```

Example (full version):

Copy the downloaded file to the destination directory `/usr/local`. You may use either OS interface or terminal to locate and copy the downloaded file:

```
> cp /your/path/to/downloaded/rubyencoder-1.1-linux.tar.gz /usr/local
```

Unpack:

```
> cd /usr/local
> tar xzf rubyencoder-1.1-linux.tar.gz
```

```
Update permissions:
> cd /usr/local/rubyencoder-1.1/bin
> chmod a+x rubyencoder licgen rginfo
```

```
Run the encoder:
> ./rubyencoder
```

or if you are using zip:

Copy the downloaded file to the destination directory /usr/local. You may use either OS interface or terminal to locate and copy the downloaded file:

```
> cp /your/path/to/downloaded/rubyencoder-1.1-linux.zip /usr/local
```

```
Unpack:
> cd /usr/local
> unzip rubyencoder-1.1-linux.zip
```

```
Update permissions:
> cd /usr/local/rubyencoder-1.1/bin
> chmod a+x rubyencoder licgen rginfo
```

```
Run the encoder:
> ./rubyencoder
```

The installation package has the following structure:

rubyencoder-1.1/bin/rubyencoder	RubyEncoder executable
rubyencoder-1.1/bin/rubyencoder18.so	Internal encoder for Ruby 1.8.x (cannot be used directly)
rubyencoder-1.1/bin/rubyencoder19.so	Internal encoder for Ruby 1.9.x (cannot be used directly)
rubyencoder-1.1/bin/license.txt	license text
rubyencoder-1.1/bin/licgen	RubyEncoder Script License Generator (for full version only)
rubyencoder-1.1/bin/rginfo	RubyEncoder Information Tool (for full version only)
rubyencoder-1.1/rgloader/*	Protected script loaders
rubyencoder-1.1/README	Startup document
rubyencoder-1.1/RubyEncoder_User_Manual.pdf	User manual in PDF format

When you run RubyEncoder for the first time please follow the [instructions below about encoder license installation](#).

2.4.2 Command line encoder installation for FreeBSD

You need to unpack the downloaded encoder installation file into any directory you wish. For FreeBSD we suggest that you install into the /usr/local or any user home directory /usr/home/username. The following instruction is expecting you use a terminal for installation. The downloaded encoder file may have the following names according to the OS and version of the encoder:

```
rubyencoder-1.1-freebsd.tar.gz (or .zip, or .tar.bz2)
rubyencoder-1.1-eval-freebsd.tar.gz (or .zip, or .tar.bz2)
```

Example (trial version):

Copy the downloaded file to the destination directory `/usr/local`. You may use either OS interface or terminal to locate and copy the downloaded file:

```
> cp /your/path/to/downloaded/rubyencoder-1.1-eval-freebsd.tar.gz /usr/local
```

Unpack:

```
> cd /usr/local
```

```
> tar xzf rubyencoder-1.1-eval-freebsd.tar.gz
```

Update permissions:

```
> cd /usr/local/rubyencoder-1.1-eval/bin
```

```
> chmod a+x rubyencoder
```

Run the encoder:

```
> ./rubyencoder
```

or if you are using zip:

Copy the downloaded file to the destination directory `/usr/local`. You may use either OS interface or terminal to locate and copy the downloaded file:

```
> cp /your/path/to/downloaded/rubyencoder-1.1-eval-freebsd.zip /usr/local
```

Unpack:

```
> cd /usr/local
```

```
> unzip rubyencoder-1.1-eval-freebsd.zip
```

Update permissions:

```
> cd /usr/local/rubyencoder-1.1-eval/bin
```

```
> chmod a+x rubyencoder
```

Run the encoder:

```
> ./rubyencoder
```

Example (full version):

Copy the downloaded file to the destination directory `/usr/local`. You may use either OS interface or terminal to locate and copy the downloaded file:

```
> cp /your/path/to/downloaded/rubyencoder-1.1-freebsd.tar.gz /usr/local
```

Unpack:

```
> cd /usr/local
```

```
> tar xzf rubyencoder-1.1-freebsd.tar.gz
```

Update permissions:

```
> cd /usr/local/rubyencoder-1.1/bin
```

```
> chmod a+x rubyencoder licgen rginfo
```

Run the encoder:

```
> ./rubyencoder
```

or if you are using zip:

Copy the downloaded file to the destination directory /usr/local. You may use either OS interface or terminal to locate and copy the downloaded file:

```
> cp /your/path/to/downloaded/rubyencoder-1.1-freebsd.zip /usr/local
```

Unpack:

```
> cd /usr/local
> unzip rubyencoder-1.1-freebsd.zip
```

Update permissions:

```
> cd /usr/local/rubyencoder-1.1/bin
> chmod a+x rubyencoder licgen rginfo
```

Run the encoder:

```
> ./rubyencoder
```

The installation package has the following structure:

rubyencoder-1.1/bin/rubyencoder	RubyEncoder executable
rubyencoder-1.1/bin/rubyencoder18.so	Internal encoder for Ruby 1.8.x (cannot be used directly)
rubyencoder-1.1/bin/rubyencoder19.so	Internal encoder for Ruby 1.9.x (cannot be used directly)
rubyencoder-1.1/bin/license.txt	license text
rubyencoder-1.1/bin/licgen	RubyEncoder Script License Generator (for full version only)
rubyencoder-1.1/bin/rginfo	RubyEncoder Information Tool (for full version only)
rubyencoder-1.1/rgloader/*	Protected script loaders
rubyencoder-1.1/README	Startup document
rubyencoder-1.1/RubyEncoder_User_Manual.pdf	User manual in PDF format

When you run RubyEncoder for the first time please follow the [instructions below about encoder license installation](#).

2.4.3 Command line encoder installation for Mac OS X

Installation for Mac OSX is easy. Double click on the downloaded DMG. When the DMG has opened, drag the RubyRncoder icon onto the Applications icon, which will install the RubyEncoder application into your Applications folder. Double click the RubyEncoder icon in Applications, then double click 'Start RubyEncoder' icon. This will start the terminal and the encoder will run in it. Follow instructions on the screen to install the RubyEncoder on your machine.

The installation package has the following structure:

rubyencoder-1.1/bin/rubyencoder	RubyEncoder executable
rubyencoder-1.1/bin/rubyencoder18.bundle	Internal encoder for Ruby 1.8.x (cannot be used directly)
rubyencoder-1.1/bin/rubyencoder19.bundle	Internal encoder for Ruby 1.9.x (cannot be used directly)
rubyencoder-1.1/bin/license.txt	license text
rubyencoder-1.1/bin/licgen	RubyEncoder Script License Generator (for full version only)

rubyencoder-1.1/bin/rginfo	RubyEncoder Information Tool (for full version only)
rubyencoder-1.1/rgloader/*	Protected script loaders
rubyencoder-1.1/README	Startup document
rubyencoder-1.1/RubyEncoder_User_Manual.pdf	User manual in PDF format

At this stage, the RubyEncoder for Mac OS X is a command line application. In order to run it you need to start the Terminal application and run `/Applications/rubyencoder-1.1/bin/rubyencoder`. Follow the [instructions below about encoder license installation](#).

2.4.4 Command line encoder installation for Windows

Installation for Windows is easy. Double click the downloaded RubyEncoder installer executable `rubyencoder-1.1-windows.exe` or `rubyencoder-1.1-eval-windows.exe` to run it. Follow instructions on screen. RubyEncoder installer will install the software and finally will run the encoder. You may double click the 'Start RubyEncoder' icon later to start the encoder from the Start menu. This will start the terminal and the encoder will run in it. Follow instructions on the screen to register the RubyEncoder on your machine.

The installation package has the following structure:

rubyencoder-1.1/bin/rubyencoder.exe	RubyEncoder executable
rubyencoder-1.1/bin/rubyencoder18.dll	Internal encoder for Ruby 1.8.x (cannot be used directly)
rubyencoder-1.1/bin/rubyencoder19.dll	Internal encoder for Ruby 1.9.x (cannot be used directly)
rubyencoder-1.1/bin/license.txt	license text
rubyencoder-1.1/bin/licgen.exe	RubyEncoder Script License Generator (for full version only)
rubyencoder-1.1/bin/rginfo.exe	RubyEncoder Information Tool (for full version only)
rubyencoder-1.1/rgloader/*	Protected script loaders
rubyencoder-1.1/README	Startup document
rubyencoder-1.1/RubyEncoder_User_Manual.pdf	User manual in PDF format

When you run RubyEncoder for the first time please follow the [instructions below about encoder license installation](#).

2.5 Running the command line encoder

RubyEncoder 1.1 is a command line executable file named 'rubyencoder'. You may find it in the RubyEncoder installation directory in the `/bin` subdirectory. Running the encoder without any options will print a list of all available options for quick help. Please refer to this user manual for details on using the encoder. In your terminal change the current directory to the RubyEncoder installation directory and type `bin/rubyencoder` to run the encoder.

2.5.1 First run

You need to read and accept the RubyEncoder 1.1 license during the first run of the encoder. The License will only be displayed on the first run. Please read it, press the Enter/Return key for the next page, and finally if you agree with the license, type "I AGREE" and press the Enter/Return key on the last page when asked. It is a requirement that you need to accept the license terms in order to continue.

If the RubyEncoder 1.1 license is accepted you will receive a web link to our site and a hexadecimal

registration code on the screen. You need to visit the following URL:

<http://www.rubyencoder.com/>

and login using the username and password we provided to the specified email account provided during your download or purchase. If the username and password are correct you will be logged into the "My Account" area. Once you have entered the system, select the hexadecimal registration code that was displayed earlier by the RubyEncoder application. Copy and paste this into the corresponding field of the "My Account" area on our site ("Enter registration key:") in the "Available licenses" section. When you have done the above, click on 'Create License' and this will generate a license. Download will start automatically. If it does not start then click on the "Download" link in the "Available licenses" section.

Save the license (encode.lic) file that is created and copy it into the command line encoder installation directory into bin/ subdirectory.

2.5.2 Usage

Run the rubyencoder executable without parameters to get a list of all available options.

```
single file:  rubyencoder [options] file.rb
multiple files: rubyencoder [options] file1.rb file2.rb file3.rb
file mask:   rubyencoder [options] *.rb
file list:   rubyencoder [options] @filelist
```

You may run the RubyEncoder 1.1 encoder to encode either one or multiple files. You may enumerate all files you want to encode or use a file mask or file list to specify multiple files. A file list is a text file with either full or relative file paths of all the files to encode, separated by a new line. You should use an @ sign before the filelist name in the command line. A file list passed to the RubyEncoder encoder for batch processing from the command line may contain file masks. Standard ? and * symbols are available.

The encoded file will replace the original file. The original file will be backed up with a .bak extension by default (until you turn off the backup facility with a -b- option).

If -o option is used to specify output directory the original file will not be backed up. Instead of it, the original file will be copied to the output directory and encoded there.

RubyEncoder will encode scripts for Ruby 1.8.x by default. Use --ruby option to specify versions of Ruby you need to encode scripts for. Available values for --ruby option is 1.8 and 1.9. To encode a script to run under Ruby 1.8.x and 1.9.x use the --ruby option twice. Please note, your script(s) should be compatible with all specified versions of Ruby, otherwise you will get an error message for this file (see below).

Example:

```
> rubyencoder file1.rb
> rubyencoder --ruby 1.8 --ruby 1.9 file2.rb file3.rb file4.rb
```

You may enumerate all the files you want to encode in a file list. A file list is a text file with either full or relative file paths of all the files to encode, separated by a new line (masks are supported, use '*' and '?' for it). You should use an @ sign before the file list name in the command line. Usage:

```
>rubyencoder @filelist
```

When specifying a relative path don't use ../ or ./ directory specifiers.

It's possible to use shorter syntax for directory encoding. All specified directories will be recognized and the "*" file mask will be added:

```
>rubyencoder -r source_dir
```

which will work as if a file mask was specified:
`>rubyencoder -r "source_dir/*"`

You will see the log file printed in the terminal window during the encoding process. The Encoding status message will be displayed for each encoded file. You may get the following status messages:

ok	The file was encoded without problem.
file not found cannot be read	The specified file could not be found. Check the specified file path.
Ruby syntax or other compiler error	The original file has syntax or other errors and thus cannot be encoded. Check your file, test it with the Ruby interpreter. This error may also appear when encoding for multiple versions of Ruby. Please note, your ruby script should be compatible with all versions of Ruby you are encoding for.
could not backup source file, skipped	The encoder could not make a backup copy of your original file (when no output directory was specified). RubyEncoder skips the file in that case to keep your original version. Check you have enough free space available and permissions to write to your original files directory.
cannot not write file	The encoder could not write the encoded file. Check you have enough free space available and permissions to write to original files directory or to the output directory if you have specified it with the -o option.
file is already processed by RubyEncoder	The encoder will not encode files which are already encoded with RubyEncoder. Check your original files directory.
empty file, skipped	The encoder will not encode empty files. If you need to have empty files for any reasons you may copy them manually.
not regular file, skipped	The encoder could not encode a file because it is not a regular file. It may be a socket or a unix device for example.
copied	The file was copied without encoding. It is possible when -f option is used to specify files to encode and -o option is used to specify output directory. All other files than specified in -f option will be copied as-is without encoding. It is useful for encoding an entire project directory when it also may contain non-Ruby files.
internal encoder error, unknown error	This is an internal problem with the encoder. Check you have enough free memory space to run the encoder and some free space on the disk. If it is not a memory problem then let us know about this error. Send an email to support@rubyencoder.com with a detailed description of the error and the command line used for running the encoder. We will investigate the problem. We may also need some additional information from you.

2.5.3 Script locking options (full version only)

Note for evaluation version users: all script locking options are disabled for the evaluation version of RubyEncoder. Although you may read below about the available options and script locking features available in the full version of the encoder.

--expire <dd/mm/yyyy>

With this option you can set an expiration date for the script. The script will not run on and after the specified date and comes with the error message: "script has expired". This option will override any

previous lock set with the --days option.

--days <nn>

You can set the script to expire in a number of days (from today). The script will not run after nn days from today and comes with the error message: "RubyEncoder Loader - the script has expired. Contact the script author about this problem. Error code [09]". This option will override any previous lock set with --expire option.

--timeserver <server, server, ...>

If you use a time lock option for your scripts you may wish to let the script get the world time from the online time service for expiry checks rather than using the server time. You may specify a list of time services during encoding.

Use --time-server option to specify time servers. You may specify multiple servers IP addresses or domain names separated with "," or ";"

The TIME protocol is used, TCP on port 37 on remote server. Also NTP protocol is used, UDP on port 123 on remote server. NTP will be tried first.

If you have used a time-server option then your script will *require* an internet connection to run. Time servers will be checked in the specified order. If no server from the list can be accessed an error message will be displayed and the script will stop execution:

"RubyEncoder Loader - the script requires an internet connection to run. The file has been encoded to run only when an internet connection is available. Setup an internet connection. Error code [20]"

It's a good idea to specify 2-3 time servers which will let your script work even if some of the time servers will be temporary down.

If you have multiple scripts included from each other and some of them were encoded with a time-server option then the script will access the time server only once for the first script for better performance and will use the time value from the time-server for the other included scripts.

A list of available time servers may be found [here](#).

Locking the script to work only online

You may also use the time-server option to lock your script to run only online. Use the time-server option as usual for this but don't specify an expiration date for the script. The script will try to access the online time service and will fail if it's not possible.

--domain <domain>

You can bind the script to a domain name. The encoder will lock the script to run only from the specified domain and all sub domains. If an attempt is made to run the script on a non-authorized domain, the following error message will be displayed: "RubyEncoder Loader - the script is not licensed to run on this machine or it is not running in a web server environment. Run this script in a web server environment. Contact the script author about this problem. Error code [02]". You may use this option more than once to specify multiple domains. This option may not be used with the --domain-encrypt option.

Domain name locking supports wildcards. You may lock to *.site.com and so the script will work for aa.site.com, bb.site.com etc. ? and * symbols are supported in the same manner as for specifying file masks.

Please note, the loader will be able to check the domain name the protected script is running under ONLY in a web server environment. The web server should provide the Ruby interpreter with one of the following environment variables; SERVER_NAME or HTTP_HOST. The Apache web server does it by default for a CGI interface. It also provides these environment variables for the FastCGI interface but it depends on the FastCGI server to provide these variables to the script. For the Nginx webserver you need to have an appropriate fastcgi_param line in the web server configuration file to have that variable passed to the Ruby script. For other web servers please look at the configuration accordingly.

As a developer you may wish to check if the SERVER_NAME or HTTP_HOST variable is available in the environment on the target server by printing ENV['SERVER_NAME'] or ENV['HTTP_HOST'] values.

Hint: use the name of the main domain in this option, not the name of any sub domain until you are sure you need to lock to a sub domain.

Example 1: --domain mydomain.com

The script will run from mydomain.com, www.mydomain.com, myname.mydomain.com etc but will NOT run from otherdomain.com, www.otherdomain.com, otherdomain.net etc.

Example 2: --domain www.mydomain.com

Script will run ONLY from www.mydomain.com. It will not run on the main domain mydomain.com and all other sub domains like myname.mydomain.com as well as other domains like otherdomain.com, www.otherdomain.com, otherdomain.net etc.

--domain-encrypt <domain>

You can Bind and encrypt to a domain name. The encoder will lock the script to run only from the specified domain. The encoder will use a specified domain name as a part of the key for encryption for the maximum protection. The loader will not be able to decrypt a script from the wrong domain and will display the error message: "RubyEncoder Loader - Loader - script checksum error. The encoded file has been modified. If the script requires a license file to run this error may be caused by an invalid license file. Install the original unmodified file or contact the script author about getting the original file or license file. Error code [12]". You may use this option ONLY ONCE in a command line. This option may not be used with the --domain option.

Be careful when using this option if you may possibly need to run your protected script from a sub domain. Example: --domain-encrypt mydomain.com will allow you to run the script ONLY from mydomain.com and not from www.mydomain.com and vice versa.

Domain name locking supports wildcards. You may lock to *.site.com and so the script (or external license) will work for aa.site.com, bb.site.com etc. ? and * symbols are supported in the same manner as for specifying file masks.

Please read the above comments about SERVER_NAME and HTTP_HOST environment variables. This is also applied for --domain-encrypt option.

--ip <x.x.x.x{/y.y.y.y}>

Bind script to an ip/mask. The encoder will lock the script to run only from the specified IP address. The specified IP address mask will be applied to the real IP address before comparing. So you may use this option to lock the script to a multiple IP if a mask is specified. If run from an IP address that is

not allowed, the script will display the error message: "RubyEncoder Loader - the script is not licensed to run on this machine or it is not running in a web server environment. Run this script in web server environment. Contact the script author about this problem. Error code [01]" You may use this option more than once to specify multiple ip/mask pairs. IP address mask 255.255.255.255 is used by default if not specified. This option may not be used with --ip-encrypt option.

Please note, the loader will be able to check the domain name for a protected script ONLY in a web server environment. The web server should provide the Ruby interpreter with one of the following environment variables SERVER_ADDR or LOCAL_ADDR. Apache web server does it by default for the CGI interface. It also provides these environment variables for the FastCGI interface, but it depends on the FastCGI server to provide these variables to the script. For the Nginx webserver you need to have an appropriate fastcgi_param line in the web server configuration file to have that variable passed to the Ruby script. For other web servers please look at the configuration accordingly.

As a developer you may wish check if SERVER_ADDR or LOCAL_ADDR variable is available in the environment on the target server by printing ENV['SERVER_ADDR'] or ENV['LOCAL_ADDR'] values.

--ip-encrypt <x.x.x.x{/y.y.y.y}>

Bind and encrypt to ip/mask. The encoder will lock the script to run only from the specified IP address. The encoder will use a specified IP address with applied mask as a part of the key for encryption for the maximum protection. The Loader will not be able to even decrypt a script from the wrong ip address and will display the error message: "RubyEncoder Loader - Loader - script checksum error. The encoded file has been modified. If the script requires a license file to run this error may be caused by an invalid license file. Install the original unmodified file or contact the script author regarding getting the original file or license file. Error code [12]". You may use this option ONLY ONCE in a command line. IP address mask 255.255.255.255 is used by default if not specified. This option may not be used with --ip option.

Please read the above comments about SERVER_ADDR and LOCAL_ADDR environment variables. This is also applied for --ip-encrypt option.

--mac <xx:xx:xx:xx:xx:xx>

You can bind a script to a LAN hardware (MAC) address. (Note, the "MAC address" name here is Media Access Control address and it is not related to Macintosh computers only. All networked computers have it). This address is unique for a networking adapter and so it may be easily used to identify a machine. A MAC address is 6 bytes long, with each byte represented in hex and separated with a ':' symbol. The encoder will lock a script to run only from the machine which has a networking adapter with the specified MAC address. If there is more than one LAN adapter installed then the script will check all of them. If an attempt is made to run a script from a machine without the correct adapter, then the script will display the error message: "RubyEncoder Loader - the script is not licensed to run on this machine. Contact the script author regarding this problem. Error code [03]" You may use this option more than once to specify multiple MAC addresses.

Hint: you may use 'ifconfig' command under Mac OSX, Linux or FreeBSD or 'route print' under Windows to get a list of installed networking adapters and known MAC addresses. On Macintosh you may find LAN hardware addresses in System Preferences/Network/Advanced/Ethernet/Ethernet ID.

Locking the script to a LAN hardware address may be the only option to lock the script to a machine when the script is not running in a web server environment or some environment variables are not available for server IP address or server domain name check. E.g. running a script as cron task or just a command line script.

--external <filename>

The script will require an external license file to run. This file may be distributed with the script or separately from it. This option gives you an opportunity to encode your script once and distribute to users with different licenses. Each license may have a different number of locks. You should specify only an external license file name here. Example: `--external script.lic` No real license file will be created for now. You should use RubyEncoder licgen tool for creating a license file for the script. When running protected scripts, and no specified license file is found, the script will come with the error message: "RubyEncoder Loader - the script requires ... license file to run. Contact the scrip author regarding getting a license file. Error code [13]" You may use this option only ONCE in a command line. This option may not be used with any other binding options.

--projid <value>

Allows you to specify a Project ID to identify your project. This is to be used with `--external` option. You should use the RubyEncoder licgen tool for creating a license file for the script with the same Project ID.

--projkey <value>

Allows you to specify the Project Key to encrypt your project. To be used with `--external` option. You should use the RubyEncoder licgen tool for creating a license file for the script with the same Project Key.

The algorithm used for locking a script to an external license file gives your scripts much stronger protection from reverse engineering, unlocking and bytecode stealing, but it also gives you the most flexible way to generate trial versions of your products and to lock scripts to your customer's machine. This is the most powerful and flexible way to protect your scripts. We recommend that you use external license files for all your script protection.

Short algorithm description

The algorithm uses the idea of two keys. The first key (Project Id) is stored in the encrypted area of the protected script and is used to decrypt an external license file. The second key (Project Key) is stored in the license file and it is used to decrypt the bytecode from the protected script.

Using this algorithm the encoder protects your product by preventing a full working copy from being created from, for example, a demo version. To decrypt and run a protected script a true license file for the full version of your product is required. Otherwise it's impossible to decrypt and run the bytecode.

Project Id and Project Key values are required if the external license protection method is chosen.

You should specify Project Id (`--projid`) and Project Key (`--projkey`) values using options in the command line for "rubyencoder" commands. Project Id and Project Key may be any words, numbers or random sequence but for security reasons these two values *should not* be calculated from each other. They should be independent. Also you should specify the *same* Project Id, Project Key pair for "licgen" command when generating a license for previously protected scripts.

Command line example:

```
>rubyencoder --external script.lic --projid "82Gi17Bn" --projkey "Az973Qq9" myscript.rb
>licgen --projid "82Gi17Bn" --projkey "Az973Qq9" --days 7 script.lic
```

If you have licenses for multiple RubyEncoder installations you may encode scripts on one machine and generate license files on another machine. The only condition is to set the same Project Id and Project Key values for your project on different machines.

S

If a script is run with an incorrect license file the following error message will appear:

```
"RubyEncoder Loader - The license file required to run this protected script is invalid. Contact the script author regarding getting a license file. Error code [06]"
```

If a script is run with a license file with the correct Project Id but an incorrect Project Key (this may be a cracking attempt or accidental modification of the license file or script) the following error message will appear:

```
"RubyEncoder Loader - Loader - script checksum error. The encoded file has been modified. If the script requires a license file to run this error may be caused by an invalid license file. Install the original unmodified file or contact the script author about getting original file or license file. Error code [12]"
```

Important Security Notice!

(!) Keep your Project Id and Project Key values secret.

(!) Remember your Project Id and Project Key. It's impossible to restore the values from somewhere if they are forgotten. They are required for generating licenses for your customers.

(!) When generating the Project Id and Project Key manually, please use independent values.

2.5.4 Advanced options

--const name=value

RubyEncoder also lets you define custom named constants during an encoding process, or within an external script license. Constant name/value pairs are stored internally in the encrypted area of the protected script or external license. They may be used for custom script locking or any other actions if you need to store a custom value in the protected script or script license file and then retrieve it from your protected Ruby code.

It is important to use quotes if your constant name or its value contains any spaces or other special symbols. You may define only one constant with each --const option but you may add as many --const options as you need into the command line.

```
>rubyencoder --const "licensed_for=Robin Hood" myscript.rb  
>licgen --const "licensed_for=Robin Hood" script.lic
```

To get a predefined constant value from the protected Ruby code use `RGLoader::get_const()` method. This method is defined in the RubyEncoder loader.

`RGLoader::get_const()` will return a predefined RubyEncoder constant value or nil if the constant with the specified name is not defined. Constant names are **case sensitive**.

There are 5 constants predefined for each protected script:

<code>RGLoader::get_const("encoder")</code>	Returns the name of the encoder "RubyEncoder"
<code>RGLoader::get_const("loader_version")</code>	Returns the loader version number
<code>RGLoader::get_const("encode_date")</code>	Returns the UNIX timestamp when the script was encoded

<code>RGLoader::get_const("license_date")</code>	Returns the UNIX timestamp when the script license was created. It's may differ from "encode_date" when external script license is used
<code>RGLoader::get_const("expire_date")</code>	Returns script expiration date as UNIX timestamp if it's defined in the script license or internally via script binding options during encoding

--catch err=function

You may add custom error handling functions which will catch script licensing errors. The Error handler should be a function which accepts two parameters:

```
error_handler( code , message )
```

You may use any name for this function. Also you may have different functions for different script errors. The first argument will contain an error code. The second one will contain a default error message. To set a custom error handler use --catch option for the rubyencoder command:

```
>rubyencoder --catch err=function myscript.rb
```

Where "err" is one of the predefined constants and "function" is an error handler function name.

Err	Code	Default message
ERR_LICENSE	01,02,03	script cannot run on this machine
ERR_EXTLICCRC	06	script license is invalid
ERR_EXPIRED	09	script has expired
ERR_EXTLIC	13	script requires license file to run
ERR_OFFLINE	20	script requires an internet connection to run
ERR_ALL	-	-

ERR_ALL is a special value to specify "one-for-all" error handler function.

The custom error handler function should be defined before an error may occur. The best place for it is in "prepend header" code as it's loaded before any license checking is done and so the error handler will be always available if defined there. But you may also define a custom error handler function in another encoded file which will be run before the script which may cause a license error. Don't put any passwords or secret data if you use a "prepend header" code for defining an error handler as this code is stored unencoded.

Example:

Encode ruby script with defined prepend header containing definition of my_err_handler() function. Please note additional back slashes (\) are used to quote special character in command line.

```
>rubyencoder -p "def my_err_handler(code,msg); printf \"My error handler caught error code %d with a message '%s'\n\", code, msg; end;" --catch ERR_ALL=my_err_handler --external script.lic --projid Fg3161jd --projkey 826Gdb31 hello.rb
```

The encoded script will have the defined code as unencoded in the beginning:

```
# RubyEncoder v1.0

def my_err_handler(code,msg); printf "My error handler caught error code %d with a
message '%s'\n",
code, msg; end;

_d = _d0 = File.expand_path(File.dirname(__FILE__)); while 1 do _f = _d +
'/rgloader/loader.rb'; bre
ak if File.exist?(_f); _dl = File.dirname(_d); if _dl == _d then raise "Ruby
script '"+__FILE__+"' i
s protected by RubyEncoder and requires the RubyEncoder loader. Please visit the
http://www.rubyenco
der.com/loaders/ RubyEncoder site to download the required loader and unpack it
into '"+_d0+"/rgload
er/' directory to run this protected script."; break; else _d = _dl; end; end;
require _f; RGLoader:
:load('AAEAAAEEoAAAAIAAAAA/y7C/NTZP8FnCp00d+uHwMSVgcicoaW6ERaCNJzTN2xah6H+6o9f2eG
RGZuRBc9AjPFIowkPt
+ZUc2o6qTmfb4Dk6gJ30sVgI20+Yzju02Wiv3pGK3UDOMw+MFuXAIp7+8AL1Yy4FLawYTxZqzQTnsCQgOF
ObFkJhkidkqAHHKdyf
Y7Fy0N1IhlqOMT1AWwOan2WIpMkbwbl3vykcvUpncSAAAAYAAAAPwjwCB2fc7tjHyItPnulhcuoPtNML
r51XrpsB9d1KJVC0+W
IrJB4Y301HCvzSDsC7dJ/Ak5LMBV/fVHQgOV2Is3B2/SgalIdFc1lJ6ImqOJGBQDpAjBya977eTnLMDwAA
AAAA=' );
```

Now a test run without a required license file:

```
>ruby hello.rb
```

```
My error handler caught error code 9 with a message 'RubyEncoder Loader - This
script has expired. Contact the script author about this problem. Error code [09]'
```

-p "code"

Prepend header code. You may put any code to be inserted BEFORE the protected scripts code. This code WILL NOT BE ENCODED. This should be syntactically correct Ruby code. Of course, it may be Ruby comments starting with a # symbol. This option is usually used for including copyrights into protected scripts. Also this option is used to put a custom error handler code into protected scripts (see --catch option). You should prepend all special characters including double quote characters with a back slash if you want to include them into the code (" becomes \ ").

Example 1:

```
>rubyencoder -p "# My protected script. Copyright by My Name" file.rb
```

Example 2:

```
> rubyencoder -p "puts \"My protected script. Copyright by My Name\";" file.rb
```

-j "code"

By default a RubyEncoder protected script will generate an exception with the following message when the RubyEncoder Loader is not installed:

Ruby script '...' is protected by RubyEncoder and requires the RubyEncoder loader. Please visit the <http://www.rubyencoder.com/loaders/> RubyEncoder site to download the required loader and unpack it

into './rgloader/' directory to run this protected script. (RuntimeError)

It is possible for you to change the default loader error behavior. This option allows you to change the default error action of the protected script if it cannot find or load an appropriate RubyEncoder Loader file. You may use any Ruby code here and it will be executed as a replacement to the default RubyEncoder loader exception. This code WILL NOT BE ENCODED. You should prepend all special characters including double quote characters with a back slash if you want to include them into the code (" becomes \"). If you need instead of printing own error message you may load the required RubyEncoder Loader with "require" directive from non-standard directory.

Example:

```
> rubyencoder -j "puts \"Loader is not found. Call 123-456-7890 for support\"; exit(1);" hello.rb
```

The encoded script will have the defined code as unencoded:

```
# RubyEncoder v1.0

_d = _d0 = File.expand_path(File.dirname(__FILE__)); while 1 do _f = _d +
'/rgloader/loader.rb'; break if File.exist?(_f); _d1 = File.dirname(_d); if _d1 == _d then
puts "Loader is not found. Call 123-456-7890 for support"; exit(1);
break; else _d = _d1; end; end; require _f;
RGLoader::load('AAEAAAEaAAAAIAAAAA/6vBQ2j8ivZCR3gVFB15
Hr/Y90fXBYLdnQDavAjb9qL66uNG0QBaN4Uk9NrcUbHzhv/WHM97yUQkyjHsid9nsMr9JiGEavcQ5tAIbH
D1/lxoxia2TOrPle4V
e8+H+3odwWqOIjks94QLEgAAAGAAAAB75w3qzOZQCmvQDuOs+G9ZVhz0GbAy1oT4injTT83Iijyj0iubOU
fKRn2+frb7QOm3dEXG
qgUtKkGzz2yaCE0T9yV1V0ZtZv4RKezGy1UJdYtDb41rK0t8S9FRLBYnwAYAAAAA');
```

Now a test run without a required RubyEncoder Loader installed:

```
>ruby hello.rb
```

```
Loader is not found. Call 123-456-7890 for support
```

Prepend header code and Loader error code may be loaded from a file

Prepend header code option -p and Loader error code option -j may load the source from a file. Use @filename as a parameter for -p or -j. We suggest to use this option if you need to add some complex code. It is easier to edit this code in a separate file than writing it in command line as an option.

Example:

```
>rubyencoder -p @prepend.rb -j @loadererr.rb file.rb
```

2.5.5 Other options

There are some other options available to pass to rubyencoder command line encoder:

- v Display version number
- h Display full options list

- l Display license information
- credits Display names of RubyEncoder developers
- w Wait for key press before exit. Allows you to check encoding log in terminal before exit.
- q Display settings and request confirmation. The encoder will display all encoding parameters and wait for a key press before any real encoding takes place. You may check all parameters and cancel if anything is not correct.
- r Recurse subdirectories. The encoder will recurse all subdirectories when searching files using a specified file mask.
IT IS IMPORTANT TO ENCLOSE FILE MASKS IN DOUBLE QUOTES. Otherwise, if file masks will not be enclosed in double quotes command line shell will expand file masks and recursion will not work as expected.
- b <ext> Set file extension for backup files (bak is default). You may change an extension used for backup copies with this option.
Example: -b old
- b- Disable backup of source files (BE CAREFUL!)
- x "mask" | @list You may specify what files or directories shall NOT be encoded. You may specify either a strict name, relative path with a directory name or a mask (with ? and/or *).
- Example:
>rubyencoder -r -x "doc/*" -x "config.rb" "*.rb"
- This will encode all *.rb files in the current directory and all directories recursively but all files in the "doc" directory and all files (and dirs if any!) named "config.rb" will not be encoded.
- You may enumerate all the files you want to exclude from encoding using a file list to specify multiple files. A file list is a text file with either full or relative file paths of all the files to encode, separated by a new line (masks are supported, use '*' and '?' for it). You should use an @ sign before the filelist name in the command line. Usage: -x @filelistname
When specifying a relative path don't use ../ or ./ directory specifiers.

`-f "mask" | @list` You may specify what files will be encoded by filenames, file masks or a file list. All other files which have been added for processing or found by expanding file masks will be copied into the output directory "as-is" without encoding. If you don't specify the `-f` option then all specified files will be encoded by default.

Example 1:

```
>rubyencoder -r -f "*.rb" -o "output_dir" ""
```

All (with recursion) `*.rb` files from the current directory will be copied and encoded into the `output_dir`. All other files from the current directory will be copied into `output_dir` as-is (unencoded).

You may specify multiple filenames or file masks with using of multiple `-f` options:

Example 2:

```
>rubyencoder -r -f "*.rb" -f "includes/*.rb" -f @myrubyfiles -o "output_dir" ""
```

If you don't specify the output directory but use `-f` option then only files specified with `-f` option will be encoded. All other files will remain unchanged.

You may enumerate all the files you want to encode in a file list. A file list is a text file with either full or relative file paths of all the files to encode, separated by a new line (masks are supported, use `*` and `?` for it). You should use an `@` sign before the filelist name in the command line. Usage: `-f @filelistname` When specifying a relative path don't use `../` or `./` directory specifiers.

`-o <output_dir>` You can specify an output directory for all encoded scripts. Source files will be unchanged if you specify an output directory different from your source scripts dir. The default backup option will be off when an output directory is specified. If you want to re-enable it, even when the output directory is specified, then use the `-b <ext>` option after the output directory option. The full directory path to the source scripts will be recreated under the output directory if the full path to the source files was specified.

Example 1: Encode all `*.rb` scripts in the current dir with recursion and put encoded files into `/home/myproject/encoded`.

```
> rubyencoder -r -o /home/myproject/encoded *.rb
```

Example 2: Encode all scripts specified in the filelist and put encoded files into `/home/myproject/encoded`. Additionally backup source scripts in source directory with `.bak` extension.

```
> rubyencoder -o /home/myproject/encoded -b bak @filelist
```

Always quote file masks. Otherwise the command line shell will replace your mask with the real file and dir names and the result may be unexpected. You should always quote file masks that specify files to encode or exclude in command line options (e.g. `"*.rb"`).

2.6 Using external script license generator (full version only)

The Script License Generator is an external tool for creating script license files. A script license file is required to run protected scripts encoded with the `--external` option.

You may find 'licgen' executable in the RubyEncoder installation directory in `/bin` subdirectory. Running the license generator without any options will print a list of all available options for quick help. Please refer to this user manual for details on using the license generator.

Using the script license is the best way of encoding if you need to distribute one script or entire project between different users but need to use different restriction options for each user. You need to encode your scripts with the `--external` option using RubyEncoder 1.1 and then create a license for each user with the RubyEncoder 1.1 Script License Generator.

Scripts encoded with the `--external` option will require an external license file to run. Protected scripts will search for the license file in the current directory and all parent directories. So you may have one license file for an entire protected project located in the top project directory.

If the protected script cannot find the specified license file it will come with the error message: "RubyEncoder Loader - the script requires ... license file to run. Contact the script author regarding getting a license file. Error code [13]"

The algorithm used for locking script to an external license file gives your scripts much stronger protection from reverse engineering, unlocking and bytecode stealing, but it also gives you the most flexible way to generate trial versions of your products and to lock scripts to your customer's machine. This is the most powerful and flexible way to protect your scripts. We recommend that you use external license files for all your script protection.

Short algorithm description

The algorithm uses the idea of two keys. The first key (Project Id) is stored in the encrypted area of the protected script and is used to decrypt an external license file. The second key (Project Key) is stored in the license file and it is used to decrypt the bytecode from the protected script.

Using this algorithm the encoder protects your product by preventing a full working copy from being created from, for example, a demo version. To decrypt and run a protected script a true license file for the full version of your product is required. Otherwise it's impossible to decrypt and run the bytecode.

Project Id and Project Key values are required if the external license protection method is chosen.

You should specify Project Id (`--projid`) and Project Key (`--projkey`) values using options in the command line for "rubyencoder" commands. Project Id and Project Key may be any words, numbers or random sequence but for security reasons these two values *should not* be calculated from each other. They should be independent. Also you should specify the *same* Project Id, Project Key pair for "licgen" command when generating a license for previously protected scripts.

Command line example:

```
>rubyencoder --external script.lic --projid "82Gi17Bn" --projkey "Az973Qq9" myscript.rb
>licgen --projid "82Gi17Bn" --projkey "Az973Qq9" --days 7 script.lic
```

If you have licenses for multiple RubyEncoder installations you may encode scripts on one machine and generate license files on another machine. The only condition is to set the same Project Id and Project Key values for your project on different machines.

If a script is run with an incorrect license file the following error message will appear: "RubyEncoder Loader - a license file required to run this protected script is invalid. Contact the script

author regarding getting a license file. Error code [06]"

If a script is run with a license file with the correct Project Id but incorrect Project Key (this may be a cracking attempt or accidental modification of the license file or script) the following error message will appear:

"RubyEncoder Loader - Loader - script checksum error. The encoded file has been modified. If the script requires a license file to run this error may be caused by invalid license file. Install original unmodified file or contact the script author regarding getting the original file or license file. Error code [12]"

Important Security Notice!

(!) Keep your Project Id and Project Key values secret.

(!) Remember your Project Id and Project Key. It's impossible to restore the values from somewhere if forgotten. They are required for generating licenses for your customers.

(!) When generating the Project Id and Project Key manually, please use independent values.

2.6.1 Usage

Run the licgen executable without parameters to get a list of all available options.

licgen [options] output.lic

output.lic - This is the name of the license file to generate. It should be the same that you used in --external option during the encode.

It is possible to run the licgen tool with only a license file name but without any other locking options specified. It will generate the license required to run your scripts encoded with --external option but no locking will be applied to the protected scripts. To enable locking with the external license file please read below about [script locking options](#). All locking options act the same as when specified for the rubyencoder executable.

2.6.2 Script locking options

All options listed below work exactly the same as similar locking options of RubyEncoder 1.1.

--expire <dd/mm/yyyy>

With this option you can set an expiration date for the script. The script will not run on and after the specified date and comes with the error message: "script has expired". This option will override any previous lock set with the --days option.

--days <nn>

You can set the script to expire in a number of days (from today). The script will not run after nn days from today and comes with the error message: "RubyEncoder Loader - the script has expired. Contact the script author about this problem. Error code [09]". This option will override any previous lock set with --expire option.

--timeserver <server, server, ...>

If you use a time lock option for your scripts you may wish to let the script get the world time from the online time service for expiry checks rather than using the server time. You may specify a list of time services during encoding.

Use --time-server option to specify time servers. You may specify multiple server IP addresses or domain names separated with "," or ";"

The TIME protocol is used, UDP on port 37 on the server. Also NTP protocol is used, TCP on port 123 on the server.

If you have used a time-server option then your script will *require* an internet connection to run. Time servers will be checked in the specified order. If no server from the list can be accessed an error message will be displayed and the script will stop execution:

"RubyEncoder Loader - the script requires an internet connection to run. The file has been encoded to run only when an internet connection is available. Setup an internet connection. Error code [20]"

It's a good idea to specify 2-3 time servers which will let your script work even if some of the time servers will be temporary down.

If you have multiple scripts included from each other and some of them were encoded with a time-server option then the script will access the time server only once for the first script for better performance and will use the time value from the time-server for other included scripts.

The list of available time servers may be found [here](#).

Locking the script to work only online

You may also use the time-server option to lock your script to run only online. Use the time-server option as usual for this but don't specify an expiration date for the script. The script will try to access the online time service and will fail if it's not possible.

--domain <domain>

You can bind the script to a domain name. The encoder will lock the script to run only from the specified domain and all sub domains. If an attempt is made to run the script on a non-authorized domain, the following error message will be displayed: "RubyEncoder Loader - the script is not licensed to run on this machine or it is running out of web server environment. Run this script in web server environment. Contact script's producer about this problem. Error code [02]". You may use this option more than once to specify multiple domains. This option may not be used with the --domain-encrypt option.

Domain name locking supports wildcards. You may lock to *.site.com and so the external license will work for aa.site.com, bb.site.com etc. ? and * symbols are supported in the same manner as for specifying file masks.

Please note, the loader will be able to check the domain name protected script is running under ONLY in web server environment. The web server should provide Ruby interpreter with one of the following environment variables SERVER_NAME or HTTP_HOST. Apache web server does it by default for CGI interface. It also provides these environment variables for FastCGI interface but it depends on FastCGI server to provide these variables to the script. For Nginx webserver you need to have an appropriate fastcgi_param line in the web server configuration file to have that variable passed to Ruby script. For other web servers please look on configuration accordingly.

As a developer you may wish check if `SERVER_NAME` or `HTTP_HOST` variable is available in environment on target server by printing `ENV['SERVER_NAME']` or `ENV['HTTP_HOST']` values.

Hint: use the name of the main domain in this option, not the name of any sub domain until you are sure you need to lock to a sub domain.

Example 1: `--domain mydomain.com`

The script will run from `mydomain.com`, `www.mydomain.com`, `myname.mydomain.com` etc but will NOT run from `otherdomain.com`, `www.otherdomain.com`, `otherdomain.net` etc.

Example 2: `--domain www.mydomain.com`

Script will run ONLY from `www.mydomain.com`. It will not run on the main domain `mydomain.com` and all other sub domains like `myname.mydomain.com` as well as other domains like `otherdomain.com`, `www.otherdomain.com`, `otherdomain.net` etc.

--ip <x.x.x.x{/y.y.y}>

Bind script to an ip/mask. The encoder will lock the script to run only from the specified IP address. The specified IP address mask will be applied to the real IP address before comparing. So you may use this option to lock the script to a multiple IP if mask is specified. If run from not allowed IP script will come with the error message: "RubyEncoder Loader - the script is not licensed to run on this machine or it is running out of web server environment. Run this script in web server environment. Contact script's producer about this problem. Error code [01]" You may use this option more than once to specify multiple ip/mask pairs. IP address mask `255.255.255.255` is used by default if not specified. This option may not be used with `--ip-encrypt` option.

Please note, the loader will be able to check the domain name protected script is running under ONLY in web server environment. The web server should provide Ruby interpreter with one of the following environment variables `SERVER_ADDR` or `LOCAL_ADDR`. Apache web server does it by default for CGI interface. It also provides these environment variables for FastCGI interface but it depends on FastCGI server to provide these variables to the script. For Nginx webserver you need to have an appropriate `fastcgi_param` line in the web server configuration file to have that variable passed to Ruby script. For other web servers please look on configuration accordingly.

As a developer you may wish check if `SERVER_ADDR` or `LOCAL_ADDR` variable is available in environment on target server by printing `ENV['SERVER_ADDR']` or `ENV['LOCAL_ADDR']` values.

--mac <xx:xx:xx:xx:xx:xx>

You can bind a script to LAN hardware (MAC) address. (Note, the "MAC address" name here is Media Access Control address and it is not related to Macintosh computers only. All networked computers have it. Although it is available for Macintosh as well). This address is unique for a networking adapter and so it may be easily used to identify a machine. A MAC address is 6 bytes long, with each byte represented in hex and separated with ':' The encoder will lock a script to run only from the machine which has a networking adapter with the specified MAC address. If there is more than one LAN adapter installed then script will check all of them. If an attempt is made to run a script from a machine without the correct adapter, then the script will display the error message: "RubyEncoder Loader - the script is not licensed to run on this machine. Contact script's producer about this problem. Error code [03]" You may use this option more than once to specify multiple MAC addresses.

Hint: you may use 'ifconfig' command under Mac OSX, Linux or FreeBSD or 'route print' under Windows to get a list of installed networking adapters and known MAC addresses. On Macintosh you may find LAN hardware addresses in System Preferences/Network/Advanced/Ethernet/Ethernet ID.

Locking the script to LAN hardware address may be the only option to lock the script to machine when the script is not running in web server environment or some environment variables are not available for server IP address or server domain name check. E.g. running a script as cron task or just a command line script.

--projid <value>

Allows you to specify Project ID to identify your project. To be used with --external option. You should use RubyEncoder licgen tool for creating a license file for the script with the same Project ID.

--projkey <value>

Allows you to specify Project Key to encrypt your project. To be used with --external option. You should use RubyEncoder licgen tool for creating a license file for the script with the same Project Key.

The algorithm used for locking a script to an external license file gives your scripts much stronger protection from reverse engineering, unlocking and bytecode stealing, but it also gives you the most flexible way to generate trial versions of your products and to lock scripts to your customer's machine. This is the most powerful and flexible way to protect your scripts. We recommend that you use external license files for all your script protection.

Short algorithm description

The algorithm uses the idea of two keys. The first key (Project Id) is stored in the encrypted area of the protected script and is used to decrypt an external license file. The second key (Project Key) is stored in the license file and it is used to decrypt the bytecode from the protected script.

Using this algorithm the encoder protects your product by preventing a full working copy from being created from, for example, a demo version. To decrypt and run a protected script a true license file for the full version of your product is required. Otherwise it's impossible to decrypt and run the bytecode.

Project Id and Project Key values are required if the external license protection method is chosen.

You should specify Project Id (--projid) and Project Key (--projkey) values using options in the command line for "rubyencoder" commands. Project Id and Project Key may be any words, numbers or random sequence but for security reasons these two values *should not* be calculated from each other. They should be independent. Also you should specify the *same* Project Id, Project Key pair for "licgen" command when generating a license for previously protected scripts.

Command line example:

```
>rubyencoder --external script.lic --projid "82Gi17Bn" --projkey "Az973Qq9" myscript.rb  
>licgen --projid "82Gi17Bn" --projkey "Az973Qq9" --days 7 script.lic
```

If you have licenses for multiple RubyEncoder installations you may encode scripts on one machine and generate license files on another machine. The only condition is to set the same Project Id and Project Key values for your project on different machines.

If a script is run with an incorrect license file the following error message will appear:
"RubyEncoder Loader - a license file required to run this protected script is invalid. Contact script's producer about getting a license file. Error code [06]"

If script is run with a license file with correct Project Id but incorrect Project Key (this may be a cracking

attempt or accidental modification of the license file or script) the following error message will appear: "RubyEncoder Loader - Loader - script checksum error. The encoded file has been modified. If the script requires a license file to run this error may be caused by invalid license file. Install original unmodified file or contact script's producer about getting original file or license file. Error code [12]"

Important Security Notice!

(!) Keep your Project Id and Project Key values secret.

(!) Remember your Project Id and Project Key. It's impossible to restore the values from somewhere if they are forgotten. They are required for generating licenses for your customers.

(!) When generating the Project Id and Project Key manually, please use independent values.

2.6.3 Advanced options

--const name=value

The RubyEncoder License Generator also lets you define custom named constants during an encoding process, or within an external script license. Constant name/value pairs are stored internally in the encrypted area of the license file. They may be used for custom script locking or any other actions if you need to store a custom value in the protected script or script license file and then retrieve it from your protected Ruby code.

Constants defined in the license file will override constants with the same names defined in the protected script.

It is important to use quotes if your constant name or its value contains any spaces or other special symbols. You may define only one constant with each --const option but you may add as many --const options as you need into the command line.

```
>licgen --const "licensed_for=Robin Hood" script.lic
```

To get a predefined constant value from the protected Ruby code use the `RGLoader::get_const()` method. This method is defined in the RubyEncoder loader.

`RGLoader::get_const()` will return a predefined RubyEncoder constant value or nil if constant with the specified name is not defined. Constants names are **case sensitive**.

There are 5 constants predefined for each protected script:

<code>RGLoader::get_const("encoder")</code>	Returns the name of the encoder "RubyEncoder"
<code>RGLoader::get_const("loader_version")</code>	Returns the loader version number
<code>RGLoader::get_const("encode_date")</code>	Returns UNIX timestamp when the script was encoded
<code>RGLoader::get_const("license_date")</code>	Returns UNIX timestamp when the script license was created. It's may differ from "encode_date" when external script license is used
<code>RGLoader::get_const("expire_date")</code>	Returns script expiration date as UNIX timestamp if it's defined in the script license or internally via script binding options during encoding

2.6.4 Other options

There are some other options available to pass to licgen tool:

-v	Display version number
-h	Display full options list
-l	Display license information
-w	Wait for key press before exit. Allows you to check the encoding log in terminal before exit.

2.7 Using information tool (full version only)

The Information Tool is an external tool for viewing protected scripts and script license files details. You may find the 'rginfo' executable in the RubyEncoder installation directory in /bin subdirectory. Running the information tool without any options will print a list of all available options for quick help. Please refer to this user manual for details on using the information tool.

You may get information about protected scripts, or an external script license. This may be useful for supporting your customers, checking scripts or licenses passed to them etc. You may know the date of parameters such as encode, expiration date, binding options etc from the protected script or script license. You may pass the encoded script name or script license as a parameter to this tool.

Script information: rginfo [options] file.rb
License information: rginfo [options] file.lic

Optionally you may need to specify a project key (--projkey), target ip (--tag-ip) and/or target domain (--tag-domain) to let the script information tool decrypt the encoded file and display the info. Also you need to specify the project id (--projid) value to decode and display the script license information.

It's possible to display script information only for scripts created with the same installation of RubyEncoder . To display script license information from an external file you need to know and specify the project id.

2.7.1 Usage

Script information: rginfo [options] file.rb

Options:

--tag-ip [x.x.x.x{/y.y.y.y}]	Target IP for decrypting. Required to view information about scripts encoded with --ip-encrypt option.
--tag-domain [domain]	Target Domain for decrypting. Required to view information about scripts encoded with --domain-encrypt option.

`--projkey [value]` Script Project Key for decrypting. Required to view full information about scripts protected with an external license file when such a file is not available. You need to specify the project key to decrypt the bytecode section and view information about the supported ruby versions and test the integrity of the bytecode.

If the protected script is locked to an external license file and this file is also available in the script's directory or parent directories then all information extracted from this external license file will be also automatically displayed.

License information: `rginfo [options] file.lic`

Options:

`--projid [value]` License Project ID for decrypting an external license file. Required to view information about external license file generated with RubyEncoder License Generator.

Other options:

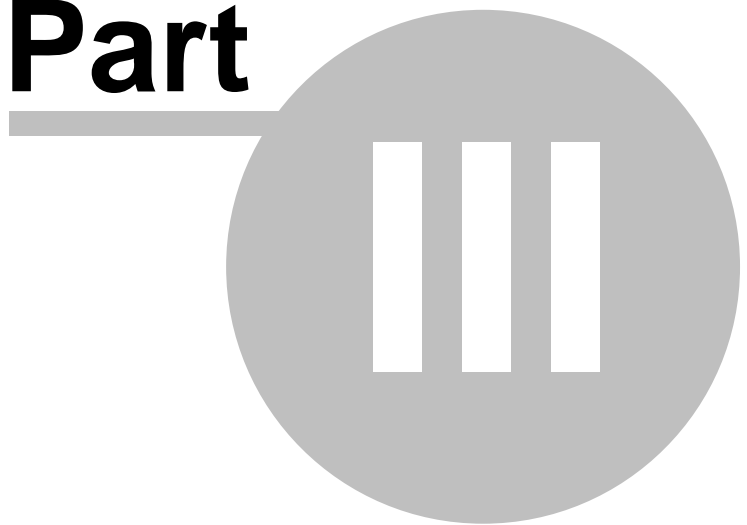
`-v` Display version number
`-h` Display full options list
`-l` Display license information

2.8 Ruby shell scripts encoding

RubyEncoder will keep the first line unchanged of the script if it begins with a `#!` UNIX shell script prefix (e.g. `#!/bin/ruby`). This lets protected scripts run from the shell or as CGI scripts. The first line of the script will not be encoded but the whole script including this line will be still protected with a checksum and so remains protected from unauthorized modifications.

RubyEncoder 1.1

Part



3 Protected script loaders

Protected script loaders are dynamically loaded Ruby modules which are automatically used by the protected script for decrypting it, validating and then running the bytecode. The source code is never restored at any time, even in memory. There is no Ruby source code within the scripts protected with RubyEncoder. There are different versions of the loaders available for different operating systems and Ruby versions. The appropriate version of the loader will be automatically loaded by the protected script.

We periodically update RubyEncoder Loaders and add support for other operating systems. The latest loaders are always freely available from our site. If you have any problems with using the loaders please make sure you always use the latest version.

<http://www.rubyencoder.com/loaders/>

RubyEncoder Loaders are included with the RubyEncoder installation package. You may find the loaders in the /rgloader subdirectory within the RubyEncoder main installation directory.

3.1 Installing loaders

Scripts protected with RubyEncoder will require installation of a RubyEncoder Loader on the target machine in order to run. Protected scripts will automatically attempt to find the loader in the rgloader/ directory located within the protected script's directory or parent directories. The code will keep looking for an rgloader/ directory with the loader.rb helper script in it until it reaches the root / folder.

To run protected scripts on a target machine you need to copy the rgloader/ directory with its full content from the RubyEncoder main installation directory to the protected script directory on a target machine. If your project consists of multiple directories, with protected scripts in them, you may copy the rgloader/ directory into your project's top directory - protected scripts will be able to find the loaders there.

If you use an FTP client for file transfer make sure you use FTP BINARY mode for uploading loader files to the target machine.

It is not required to upload loaders for all supported platforms. You may upload loader.rb helper script and one required loader. See the [next section](#) for details about naming the loader files.

3.2 rgloader directory structure

The following provides an overview of the rgloader/ directory which is a required location for RubyEncoder Loaders and file naming conventions:

loader.rb	This file is a Ruby script which helps to automatically identify the required loader. This is a required file for installation on a target machine where the protected ruby scripts will run. It is open and not encoded but you do not need to change it.
rgloader.darwin.bundle	Binary RubyEncoder Loader for Ruby 1.8.x for Mac OSX platform. It is a universal binary file compatible with i386 and ppc7400 architectures.
rgloader19.darwin.bundle	Binary RubyEncoder Loader for Ruby 1.9.x for Mac OSX platform. It is a universal binary file compatible with i386 and ppc7400 architectures.
rgloader.linux.so	Binary RubyEncoder Loader for Ruby 1.8.x for Linux i386 platform. It is 32-bit ELF shared object.

rgloader19.linux.so	Binary RubyEncoder Loader for Ruby 1.9.x for Linux i386 platform. It is 32-bit ELF shared object.
rgloader.linux.x86_64.so	Binary RubyEncoder Loader for Ruby 1.8.x for Linux x86_64 platform. It is 64-bit ELF shared object.
rgloader19.linux.x86_64.so	Binary RubyEncoder Loader for Ruby 1.9.x for Linux x86_64 platform. It is 64-bit ELF shared object.
rgloader.freebsd.so	Binary RubyEncoder Loader for Ruby 1.8.x for FreeBSD 6.x i386 platform. It is 32-bit ELF shared object.
rgloader19.freebsd.so	Binary RubyEncoder Loader for Ruby 1.9.x for FreeBSD 6.x i386 platform. It is 32-bit ELF shared object.
rgloader.freebsd.x86_64.so	Binary RubyEncoder Loader for Ruby 1.8.x for FreeBSD 6.x x86_64 platform. It is 64-bit ELF shared object.
rgloader19.freebsd.x86_64.so	Binary RubyEncoder Loader for Ruby 1.9.x for FreeBSD 6.x x86_64 platform. It is 64-bit ELF shared object.

We will include more loaders later to support other operating systems and platforms. Visit <http://www.rubyencoder.com/loaders/> for an update.

RubyEncoder 1.1

Part



4 Errors, common mistakes and possible solutions

This section includes common mistakes that people may make, either in encoding and protecting their files, or in uploading or running these files. They are not in any particular order, but we would suggest that you look at this section before you contact us regarding any support matter.

4.1 Encoded scripts modification

Encoded scripts are protected against modification. Please **DO NOT MODIFY** any single byte in the encoded scripts or you will get an error executing them.

4.2 Error messages during encoding

You will see the log file printed in the terminal window during an encoding process. Encoding status message will be displayed for each encoded file. You may get the following status messages:

ok	The file was encoded no problem.
file not found or cannot be read	The specified file could not be found. Check the specified file path.
Ruby syntax or other compiler error	The original file has syntax or other errors and thus cannot be encoded. Check your file, test it with the Ruby interpreter. This error may also appear when encoding for multiple versions of Ruby. Please note, you ruby script should be compatible with all versions of Ruby you are encoding for.
could not backup source file, skipped	The encoder could not make a backup copy of your original file (when no output directory was specified). RubyEncoder skips the file in that case to keep your original version. Check you have free space available and permissions to write to your original files directory.
cannot not write file	The encoder could not write the encoded file. Check you have free space available and permissions to write to the original files directory or to the output directory if you have specified it with -o option.
file is already processed by RubyEncoder	The encoder will not encode files which are already encoded with RubyEncoder. Check your original files directory.
empty file, skipped	The encoder will not encode empty files. If you need to have empty files for any reasons you may copy them manually.
not regular file, skipped	The encoder could not encode a file because it is not a regular file. It may be socket or a unix device for example.
copied	The file was copied without encoding. It is possible when the -f option is used to specify files to encode and -o option is used to specify output directory. All other files than specified in -f option will be copied as-is without encoding. It is useful for encoding an entire project directory when it also may contain non-Ruby files.

internal encoder error,
unknown error

This is an internal problem with encoder. Check you have enough free memory space to run the encoder and some free space on the disk. If it is not a memory problem then let us know about this error. Send an email to support@rubyencoder.com with a detailed description of the error and the command line used for running the encoder. We will investigate the problem. We may also need some additional information from you.

4.3 Error messages during protected scripts run

This section includes descriptions of error messages when running protected scripts. They are not in any particular order, but we would suggest that you look at this section before you contact us regarding any support matter.

Ruby script '...' is protected by RubyEncoder and requires the RubyEncoder loader. Please visit the <http://www.rubyencoder.com/loaders/> RubyEncoder site to download the required loader and unpack it into '.../rgloader/' directory to run this protected script. (RuntimeError)

RubyEncoder Loaders are not installed or have been installed in the wrong directory or the script does not have enough permissions to access the rgloader/ directory of files in it. To run protected scripts on a target machine you need to install the RubyEncoder Loaders to the rgloader/ directory within the protected script directory on a target machine. If your project consists of multiple directories with protected scripts in them you may install the rgloader/ directory into a project's top directory - protected scripts will be able to find the loaders there.

**The RubyEncoder loader is not installed. Please visit the <http://www.rubyencoder.com/loaders/> RubyEncoder site to download the required loader for '...your OS name...' and unpack it into '.../rgloader' directory to run this protected script.
in `require': no such file to load -- ... (LoadError)**

The protected script was able to locate the rgloader/ directory and run the loader.rb helper script from it, however the loader required for your platform was not found. Please [check](#) if your platform is supported and install an appropriate loader file to the rgloader/ directory you have.

RubyEncoder Loader - This script is not licensed to run on this machine or it is running out of web server environment. Run this script in web server environment. Please contact the script author about this problem. Error code [01]

The protected script was encoded with the --ip locking option and has bound the script to some IP address(es). When the script is run on a web server running a different IP address the above error message will appear. Check the IP address of the machine running the script. Check the script is running in a web server environment and the server IP address is available as an environment variable for the script. See [script locking options section](#) in this manual for details.

RubyEncoder Loader - This script is not licensed to run on this machine or it is running out of web server environment. Run this script in web server environment. Please contact the script author about this problem. Error code [02]

The protected script was encoded with a --domain locking option and has bound the script to some

domain name(s). When the script is about run on a web server running a different domain the above error message will appear. Check the domain name in the URL accessing the script. Check the script is running in a web server environment and that the server domain name is available as an environment variable for the script. See [script locking options section](#) in this manual for details.

RubyEncoder Loader - This script is not licensed to run on this machine. Please contact the script author about this problem. Error code [03]

The protected script was encoded with a --mac locking option and has bound the script to some LAN hardware MAC address(es). When the script is about run on a machine running a different MAC address the above error message will appear - MAC address(es) specified for the script during encoding do not match any of MAC addresses on a target machine. Check the MAC address(es) on the machine running the script. We suggest to use "ifconfig -a" command for it. Execute it in shell on a target machine to list all available network interfaces. See [script locking options section](#) in this manual for details.

RubyEncoder Loader - A license file required to run this protected script is invalid. Contact the script author for getting a license file. Error code [06]

The protected script was encoded with an --external locking option and bound to an external license file. That license file is required to run the protected script on a target machine. The license file has been found but it failed validation. Possibly it was accidentally changed. Install the correct license file for this project or generate a new license file with the same Project ID and Project Key values that were used for encoding the script. Refer to the [script locking options](#) section in this user manual for details.

RubyEncoder Loader - This script does not support version ... of Ruby. Please contact the script author about this problem. Error code [07]

RubyEncoder supports encoding for multiple versions of Ruby. You need to use the --ruby option for this in the rubyencoder command. The above message says that the script was not encoded for the version of Ruby currently running the script. This may happen, for example if you encode the script for Ruby 1.8 (--ruby 1.8 option) and try to run it under Ruby 1.9. Make sure the script is encoded for the version of Ruby on your target server and that it is compatible with that version of Ruby (otherwise you will get an error message during encoding). Please note, if you do not specify --ruby option in rubyencoder command, your scripts will be encoded only for Ruby 1.8.x (which is the latest stable version at the moment of writing this manual).

RubyEncoder Loader - This script has expired. Please contact the script author about this problem. Error code [09]

The protected script was encoded with an expiration date set for the script or external license file. The above message says that the period of using the script has expired and you cannot use this script anymore. Examples of this use are for creating demo versions of for time-limited licensing of products.

RubyEncoder Loader - Script ... header is broken. The encoded file has been modified. Install the original unmodified file or contact the script author for getting the original file. Error code [10]

RubyEncoder Loader - Script ... is broken. The encoded file has been modified. Install the original unmodified file or contact the script author for getting the original file. Error code [11]

RubyEncoder Loader - Script ... loader checksum error. The encoded file has been modified. Install the original unmodified file or contact the script author for getting the original file. Error code [17]

RubyEncoder Loader - Decompression error status The encoded file has been modified. Install the original unmodified file or contact the script author for getting the original file. Error code [18]

The RubyEncoder Loader performs a number of validation actions for the protected script before it runs the script. The above message says that some validation has failed. It may happen because the script has been changed, possibly accidentally. Any changes to the protected script are restricted for security reasons. Nobody is allowed changes in a protected script - even its author. Do not change the protected script, install original file or re-encode the script again. Unprotected areas of the script (shell command, loader code, your custom header code or custom error code) are still protected with CRC and should not be changed. Different messages above indicate different validation stages that the loader performs for the protected script.

RubyEncoder Loader - Script ... checksum error. The encoded file has been modified. If this script requires a license file to run this error may be caused by an invalid license file. Install the original unmodified file or contact the script author for getting the original file or license file. Error code [12]

This error message may be caused by the same reasons as two error messages above. Additionally this error may happen when the protected script was locked to an external license file but the wrong license file was installed on a target machine. I.e. the license file was found by name but it is incorrect, generated for a different project or just broken. Thus decryption of the protected script bytecode failed. It is important to use the same Project ID and Project Key values for the license file that were used for encoding the script. Refer to the [script locking options](#) section in this user manual for details. Do not change the protected script - install the original file or re-encode the script again. If an external license file is used - install correct license file for this project or generate a new license file with the same Project ID and Project Key values that were used for encoding the script.

RubyEncoder Loader - This script requires ... license file to run. Contact the script author for getting a license file. Error code [13]

Protected script was encoded with --external locking option and bound to an external license file. That license file is required to run the protected script on a target machine. Name of the license file should match the name specified during encoding. Use RubyEncoder License Generator to create a license file and Install it on a target machine into the script's directory or any parent directory. If the license file is already there - check if the script has enough permissions to read the license file. It is important to use the same Project ID and Project Key values for the license file that were used for encoding the script. Refer to [using scriptlicense generator](#) section in this user manual for details.

RubyEncoder Loader - Evaluation loader has expired. This file has been encoded with evaluation version of RubyEncoder. Please download and install <http://www.rubyencoder.com/loaders/> latest loaders. Error code [14]

The RubyEncoder Loader you installed on a target machine is expired. This may happen only with the scripts encoded with demo version of RubyEncoder. Download and install updated loaders from <http://www.rubyencoder.com/loaders/>

RubyEncoder Loader - Incompatible loader version. The file has been encoded with a newer version of RubyEncoder. Please download and install <http://www.rubyencoder.com/loaders/>

latest loaders. Error code [19]

Your protected script has been encoded with a newer version of RubyEncoder than the installed Loader. Download and install updated loaders from <http://www.rubyencoder.com/loaders/>

RubyEncoder Loader - This script requires an internet connection to run. The file has been encoded to run only when an internet connection is available. Setup an internet connection. Error code [20]

The protected script was encoded with --time-server option or this option was specified for the external license file the script is bound to. The RubyEncoder Loader is trying to connect to the specified time server(s) using TIME or NTP protocols. If it is not possible to connect, for any reason (connection lost, error etc) the above error message will appear. Check that the time servers specified for the protected script or script license file exist. Use the RubyEncoder Information tool to know what time servers (or any other options) were specified during encoding. We suggest that you specify a number of available time servers so your script can run even if some time servers are temporary down. See the [script locking options](#) section in this user manual for details.

RubyEncoder Loader - Insufficient memory. Error code [FF]

This may happen if the RubyEncoder Loader failed to allocate some memory (RAM) using standard library functions. Check your system has enough free memory for running the protected script. RubyEncoder Loader does not need much free memory available, so this error will usually never appear. Otherwise your system cannot work anyway without free memory available.

RubyEncoder Loader - Internal error: ...

Something wrong happen during the protected script execution. If you see this error please contact support to let us know about it support@rubyencoder.com.

Based on our experience in supporting RubyEncoder we must say that most protected scripts errors are caused by three factors: absence of loaders, modifications in the protected script or a script license or action of locking options used during the encoding process. As a final note we suggest you to use RubyEncoder Information Tool to know details about locking options for your scripts and script license files. See [using information tool](#) section in this user manual for details.

Index

- A -

About RubyEncoder™ 2
Advanced options 18, 29

- B -

Backup 21
Binding 13, 25
Buy 2

- C -

Catch errors 18
CGI 31
Command line encoder installation 6
Command line encoder installation for FreeBSD 8
Command line encoder installation for Linux and FreeBSD 7
Command line encoder installation for Mac OSX 10
Command line encoder installation for Windows 11
Cross platform encryption 6
Custom constant 18, 29
Custom header code 18

- E -

encode.lic 11
Encoded scripts modification 36
Encoder errors 36
Error 36
Error handler 18
Error messages during encoding 36
Error messages during protected scripts run 37
Errors, common mistakes and possible solutions 36
Evaluation scripts 13, 25
Exclude files 21
External license file 13, 24, 25

- F -

Features 2
First run 11
FreeBSD 7, 8

- H -

How it works 2
How to buy 2

- I -

Install 33
Installation 6, 7, 8, 10, 11
Installing loaders 33

- L -

License 11, 21, 30
licgen 24, 25, 29, 30
Linux 7
Loader error 18
Loader errors 36, 37
Loaders 33
Locking 2, 13, 25
Locking to domain 13, 25
Locking to IP 13, 25
Locking to MAC address 13, 25

- M -

Mac 10

- O -

Options 2
OSX 10
Other options 21, 30
Output directory 21

- P -

Project ID 13, 25

Project Key 13, 25
Protected script loaders 33
Purchase 2

- R -

rginfo 30
rgloader directory structure 33
Ruby 6
Ruby shell scripts encoding 31
rubyencoder 11, 12, 13, 18, 21, 25
Running the command line encoder 11

- S -

Script locking options 13, 25
Setup 6, 7, 8, 10, 11, 33
Supported OS and platforms 33
Supported Ruby versions 6

- T -

Time server 13, 25

- U -

Ultimate Ruby and Ruby on Rails Scripts Protection
6
Usage of Information Tool 30
Usage of License Generator 25
Usage of RubyEncoder 12
Using external script license generator 24
Using information tool 30

- V -

Version 21, 30
Versions 6

- W -

Windows 11