



USER'S GUIDE

Version 1.1

© 2009 algorithmprotection.com



Table of Contents

1.	INTRODUCTION	3
1.1	GENERAL INFORMATION	3
1.2	OVERVIEW OF TAURUS.....	3
2.	PASSWORD PROTECTION	5
3.	LICENSE KEY FILE PROTECTION	6
3.1	USER INDEPENDENT KEY	7
3.2	TIME DEPENDENT KEY	10
3.3	USER DEPENDENT KEY.....	15
3.4	USER AND COMPUTER DEPENDENT KEY.....	19
3.5	USER AND COMPUTER AND TIME DEPENDENT KEY	24
4	REGISTER PRODUCT MENU	29
5	FILE ENCRYPTION	32
6	GENERAL GUIDELINES.....	33
7	PRACTICAL EXAMPLES OF CONTROLLED LICENSE KEY DISTRIBUTION	34
7.1	SELLING SOFTWARE ONLINE.....	34
7.2	COPING WITH HEAVY INFORMATION SECURITY NEEDS	35

1. INTRODUCTION

1.1 GENERAL INFORMATION

Thank you for choosing Taurus® as your proprietary MATLAB® application code protection tool. This guide will demonstrate all the protection and licensing alternatives contained in Taurus® as well as provide suggestions for implementing common code features - such as limited functionality trials - through license checks. Note that Taurus® LITE does not contain all of the copy protection schemes described in this document. Also note that neither Taurus® LITE nor Taurus® STANDARD support encryption of data files as described in this document. The sections describing individual protection schemes are written in a self-contained way for easier readability; this choice inevitably results in certain repetition throughout the document.

In order to run Taurus® in the MATLAB® environment, you must have a valid license file (e.g. `taurus_license.dat`) in the same directory as the main file (e.g. `Taurus.p`), which is executed by typing the file name (e.g. `Taurus`) in the MATLAB® command prompt.

You are invited to take a look at the code examples included in the Taurus® package for hands-on information.

If you experience problems protecting your m-files with Taurus®, please contact:

support@algorithmprotection.com

We kindly remind you that if you have purchased Taurus® LITE version, then you are entitled to limited customer support. If you have purchased Taurus® STANDARD or Taurus® PROFESSIONAL, then you are entitled to full customer support. Please inform us which one of these you are using in the first e-mail, and also include your purchase receipt number.

If you request customization of Taurus® features, for a project quotation please contact:

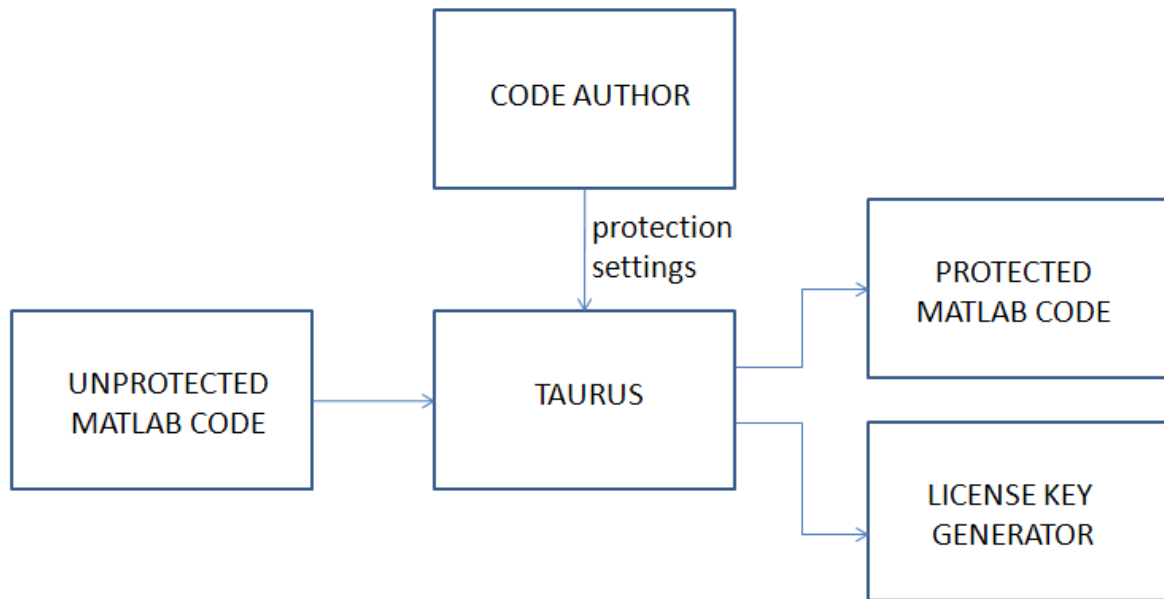
customize@algorithmprotection.com

If you encounter any problems with Taurus® license keys, please contact:

license@algorithmprotection.com

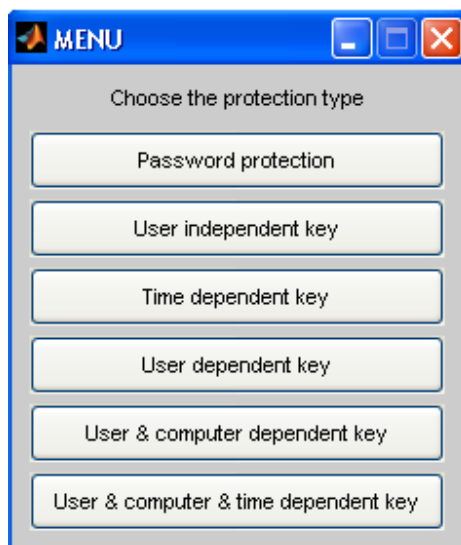
1.2 OVERVIEW OF TAURUS

The basic procedure of protecting your MATLAB code with Taurus is as follows:



Taurus® parses the unprotected m-file, and, based on certain input from the code author, generates the protected p-file as well as a license key generator GUI whenever appropriate. The MATLAB® pcode functionality obfuscates the m-file, whereas Taurus implements the chosen copy protection framework. **Neither the key generator nor the protection settings (i.e. the root key) should ever be distributed to the end users of your application;** instead, the code author or a certified license key issuer shall use the key generator to generate valid license files for the users of the application.

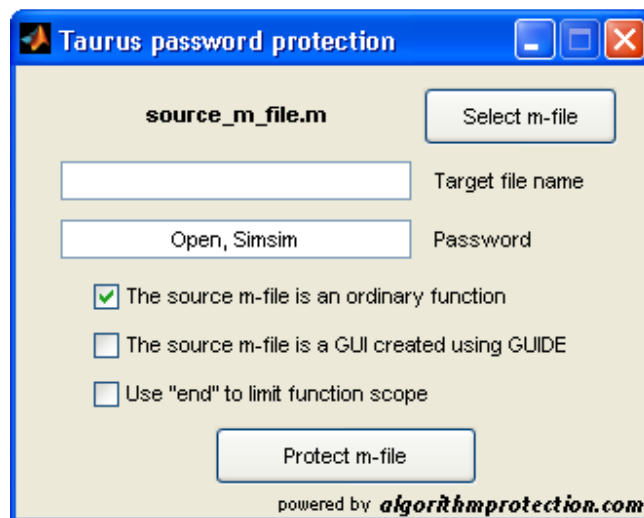
When you run Taurus® at MATLAB® command prompt, you will be asked to choose the protection type in the main menu (the menu only contains the first two items in Taurus® LITE):



Choose “Password protection” to implement a single password protection in your m-file. The other choices in the main menu will have you implement various license file checks, and the associated key generators as described above. The details of these choices are explained in the sections to follow.

2. PASSWORD PROTECTION

The password protection features are very simple to use. After clicking “Password protection” in the main menu, you will be shown the following interface:



In this menu:

- The source m-file to be protected is selected by clicking “Select m-file”.
- The name of the output file is typed in the “Target file name” edit box. Note that the target file name cannot be the same as the source file name (for otherwise the original file would be overwritten).
- The required password should be typed in the “Password” edit box. The password is case-sensitive and can contain numbers and other ASCII characters.
- Either “The source m-file is an ordinary function” or “The source m-file is a GUI created using GUIDE” should be checked, whichever is appropriate for the application to be protected.
- If the source file uses nested functions (or for other reasons the built-in functions conclude with the string “end”), the third check box in the menu should be checked. Otherwise the checkbox is left unchecked.
- Finally, the source file is protected by clicking “Protect m-file”.

In the case displayed above, the password protected file would be called “target_file.p” and the user would be required to type in the password “Open, Simsim” in order to run the function.

NOTE: By default, the correct password will be retained in the memory so that the user will not have to re-type it every time the protected function is called. You can disable this feature by including the line

```
eval('clear Taurus_passcomp');
```

at the end of your main function (or the opening function if a GUI is being protected). You have to use "eval" as above; otherwise the protection may fail.

3. LICENSE KEY FILE PROTECTION

Taurus LITE facilitates one license key copy protection scheme, whereas Taurus STANDARD and Taurus PROFESSIONAL provide five (5) different schemes for license key file protection. In the order of increasing complexity they are as follows.

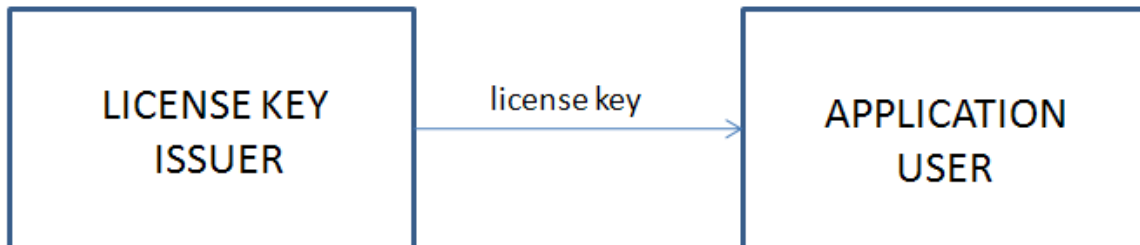
- **USER INDEPENDENT KEY:** In this case, there is a (potentially large) number of pre-built license keys for your application, each of which is assigned to at least one legal copy of the application. Since the license keys are pre-built according to a mathematical "scrambling rule", in this protection framework the license keys do *not* contain any information on the authorized users' identities, the computers on which the application is run and on time. On the other hand, because of its simplicity, the USER DEPENDENT KEY protection framework is very well suitable for automated key distribution systems. Indeed, the entire key set can be generated by just one run of the key generator and the resulting keys can be stored in a single file on a web server. They can then be easily configured to be available one by one for authorized users.
- **USER DEPENDENT KEY:** This is the simplest of those protection features in Taurus® in which certain details of the authorized user's name and organization are encrypted into the license file. However, in this protection framework, there is no dependence on time nor the computer on which the protected application is run. While a lightweight alternative for individual licensing, this feature is most suitable for "corporate licenses", whereby each member of an organization is able to run the application. The name of the licensed user as well as his/her organization can be displayed in any application protected using this framework. Displaying the name and organization of the licensed user has been observed to help prevent unauthorized distribution of software.
- **TIME DEPENDENT KEY:** In this protection framework, Taurus® includes information on the number of days the license is valid from the date of issuing into the license file. Depending on the choice of the license issuer, there may also be a dependence on user's identity, but there is no dependence on the computer on which the application is run. Once protected in this framework, the application is not allowed to be executed if the license has expired. This license protection framework is particularly suitable for distinguishing between a limited duration trial

version and a full version of the application. For example, Taurus® itself is copy protected using this framework.

- **USER & COMPUTER DEPENDENT KEY:** This protection framework is the same as the user dependent key, but, in addition, details of the computer on which the application is run are encrypted into the license file. In this case the application can only be run by a specific person on a specific computer. Obviously, this protection framework is best suited for such applications where the distribution of working copies must be strictly at the control of the license issuer (e.g. as in defense industry).
- **USER & COMPUTER & TIME DEPENDENT KEY:** This is the same as the USER & COMPUTER DEPENDENT KEY framework, but in this case the license file is also time-dependent. In other words, the license is valid for a pre-specified amount of days after which the application can only be started with a renewed license. This is the strictest form of protection available in Taurus, as the distribution of the licensed product can be confined to selected users and computers. In addition, the impact of security breaches (e.g. in the form of a stolen applications) can be controlled by controlling the duration of validity of each license key.

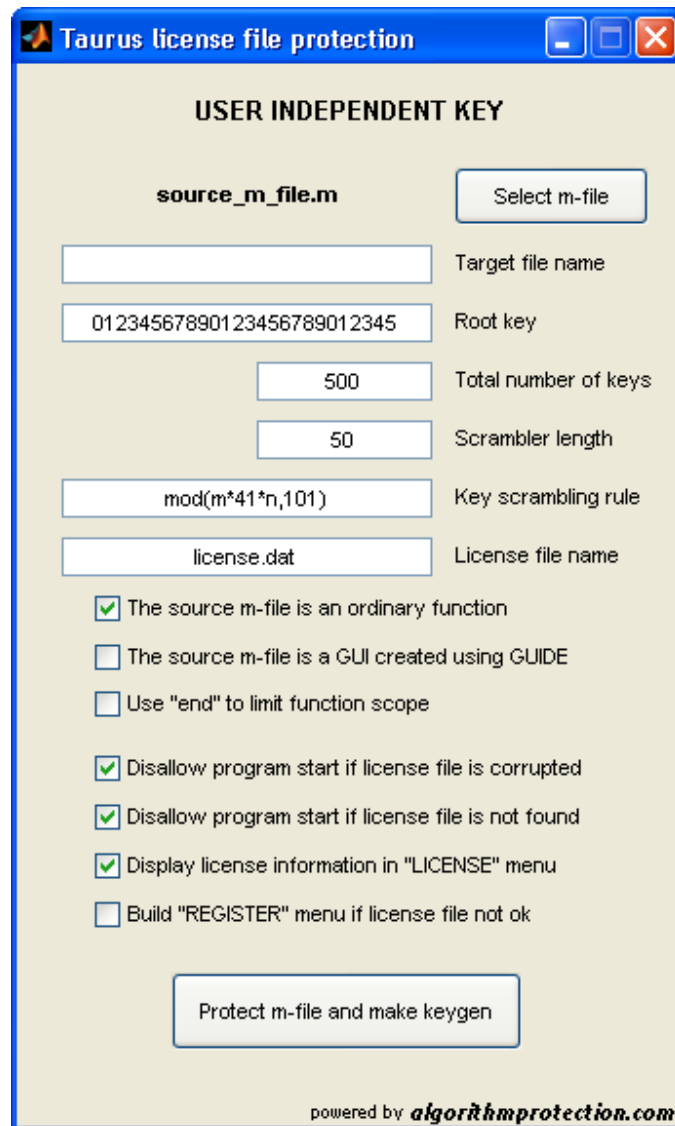
3.1 USER INDEPENDENT KEY

The procedure for user independent license key use is as follows.



Once the license key issuer has generated a license key for the application user (using the key generator), no further interaction between these two parties is necessary for running the application.

The user independent license key protection is implemented in Taurus® by choosing "USER INDEPENDENT KEY" in the main menu. The below menu then appears.



In this menu:

- The file to be protected is selected using “Select m-file” button.
- The target file name is entered in the “Target file name” edit box.
- The root key is a **secret** encryption key consisting of standard ASCII set characters (excluding some of them, like ‘). While there are only few specific requirements for the root key, it should be as “random” a string as possible. **It is equally important that the root key is not distributed to the end users of the protected software** (for otherwise the users will potentially be able to bypass the copy protection).
- The total number of keys (500 in the above screen shot) is the number of times the key scrambling rule (“mod(m*41*n,101)” in the above screen shot) is executed to generate a new

key. Once protected, your application will check if the decoded license matches any one of these keys. In practice, 500 keys may very well suffice for various applications because the user does not know if the key is unique or not.

- In the key scrambling rule, m and n are positive integers, and the end result of the mathematical scrambling rule should be a positive integer, too. While there are only few other constraints on the scrambling rule, it must be evaluable (through the “eval” command) by MATLAB®.
- The scrambler length corresponds to the key length. The longer the key, the more securely copy protected your application is. The scrambler length is also one of the two parameters in the scrambling rule.
- The license file name corresponds to the file name from which your protected application attempts to read the license key.
- The ordinary function / GUI checkboxes as well as the function scope limiter string checkbox work similarly as in the PASSWORD PROTECTION scheme.

The most important features for m-file protection are the “disallow program start if license file is corrupted” and “disallow program start if license file is not found” checkboxes. By checking both of them, one can ensure that the application can only be run by a license holder. The last two checkboxes (“Display license information in “LICENSE” menu / “Build “REGISTER” menu if license file not ok”) can be used to implement pre-built functionality for registering an unregistered product and displaying license information. The “REGISTER” functionality is described later in this guide. On the other hand, including “LICENSE” information in this protection framework may result in confusing application behavior since the licenses do not contain any user-specific information in the USER INDEPENDENT KEY protection framework. Note that you can customize these features, as the presence of a valid of license key file is exposed to your main function as described below.

By un-checking the two “disallow program start” checkboxes, it is also possible to implement a free trial, or demo version, of your software. To do so, place the following code at the start of your main function (or opening function if running a GUI made using GUIDE):

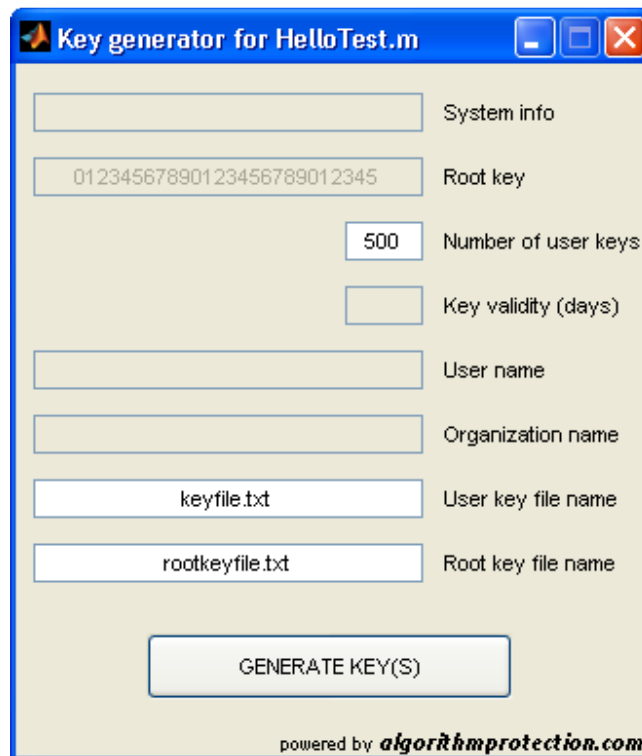
```
free_trial_mode = 1;
try
    eval('free_trial_mode = isempty(registered_user);');
catch
end
```

The variable “free_trial_mode” will be given value 1 by Taurus® protection algorithms during the launch of your application provided that the user does not have a valid license or if the license file could not be found. In the remaining parts of your application, you can enable/disable the features of your choice by comparing the value of “free_trial_mode” to 1. You should end your main function (or opening function if GUI) by clearing the persistent variables using:

```
clear functions
```

IMPORTANT REMARK: In order to speed up reoccurring calls, by default, applications protected using Taurus are set to remember a successful license file read. You can reverse this feature easily by issuing “clear functions” at the end of your main function (or opening function if GUI).

The key generator is created automatically during the protection of the m-file. In this case the key generator consists of a graphical user interface as follows:

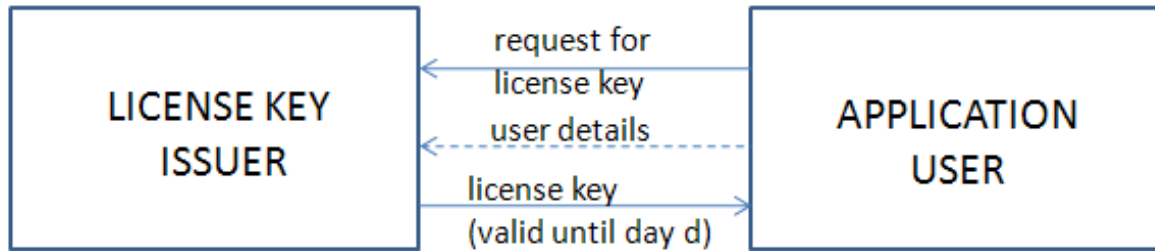


By clicking GENERATE KEY(S) above, the key generator would create 500 keys (built according to the rules above and using the specified root key) and saves them to “keyfile.txt”. Each of the 500 rows corresponds to a valid license. In the above case, authorized users should obtain a file called “license.dat”, which contains precisely one row of “keyfile.txt”.

The root key details are saved for future reference in “rootkeyfile.txt”. This is useful if you have different versions of your application, each using a different root key for encryption. The root key file should never be distributed to the end users of your application.

3.2 TIME DEPENDENT KEY

The procedure for time dependent license key use is as follows.



If the prospective application user does not have a valid license key, he/she must somehow obtain it from the license key issuer, i.e. request for a key. The key request may or may not contain information on the prospective user's identity, depending on the protected application. Based on the license configuration of the requestor, the key issuer will choose in the key generator the number of days a license key will be valid, and submit a new license file to the application user. Information on the user's identity as well as license expiry is encrypted into the license file. This process is repeated every time a license key has expired.

Time dependent license key protection is implemented in Taurus® by choosing "TIME DEPENDENT KEY" in the main menu. The below menu then appears.

Taurus license file protection

TIME DEPENDENT KEY

source_m_file.m Select m-file

Target file name

Root key

Total number of keys

Scrambler length

Key scrambling rule

License file name

The source m-file is an ordinary function

The source m-file is a GUI created using GUIDE

Use "end" to limit function scope

Disallow program start if license file is corrupted

Disallow program start if license file is not found

Display license information in "LICENSE" menu

Build "REGISTER" menu if license file not ok

Protect m-file and make keygen

powered by algorithmprotection.com

In this menu:

- The file to be protected is selected using "Select m-file" button.
- The target file name is entered in the "Target file name" edit box.
- The root key is a **secret** encryption key consisting of standard ASCII set characters. While there are only few specific requirements for the root key, it should be as "random" a string as possible. **It is equally important that the root key is not distributed to the end users of your application** (otherwise they will potentially be able to bypass the copy protection).
- The total number of keys and scrambler length are not relevant for this protection framework, and cannot be changed.
- License file name corresponds to the file name from which your protected application attempts to read the license key.

- The ordinary function / GUI checkboxes as well as the function scope limiter string checkbox work similarly as in password protection.

The most important features for m-file protection are the “disallow program start if license file is corrupted” and “disallow program start if license file is not found” checkboxes. By checking both of them (default), one can ensure that the application can only be run by any holder of a valid license key. We recommend that you experiment with the functionality of these checkboxes to ensure a proper functionality for your application before releasing it.

If you want your application to display license information in a pre-designed information box, check “Display license information in “LICENSE” menu”. If you want Taurus to implement built-in features for requesting renewal of an expired license etc., check “Build “REGISTER” menu if license not ok”. Note that in this protection framework these features can be built into the application without Taurus, too.

If you want to use information on the number of days until license expiry in your own code, place the following lines at the beginning of your main function (or opening function if protecting a GUI made using GUIDE):

```
license_days_left = 0;
try
    eval('license_days_left = free_trial_days_left;');
catch
end
```

The variable `license_days_left` will contain the number of days left contained in the license key.

If you want to use the name of the licensed user in your own code, place the following lines at the beginning of your main function (or opening function if protecting a GUI made using GUIDE):

```
registered_to = '';
try
    eval('registered_to = registered_user;');
catch
end
```

The variable `registered_to` will be assigned the user name encrypted into the license file. If a valid license can be read from a file, this variable is not empty after the above try-catch structure.

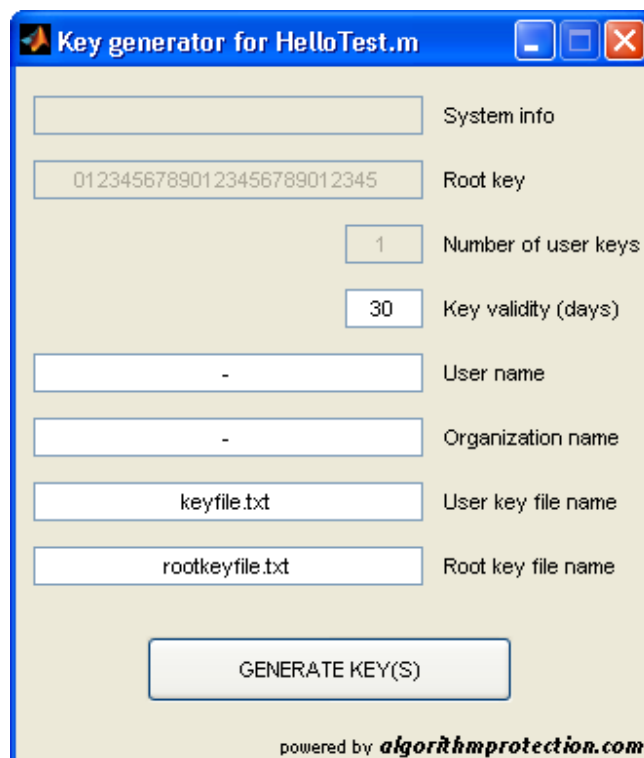
If you want to use the name of the organization of the licensed user in your own code, place the following lines at the beginning of your main function (or opening function if protecting a GUI made using GUIDE):

```
registered_organization = '';
try
    eval('registered_organization = registered_company;');
catch
end
```

The variable `registered_organization` will be assigned the organization name encrypted into the license file. If a valid license can be read from a file, this variable is not empty after the above try-catch structure.

REMARK: The above variables remain empty and `license_days_left` remains 0 if a license file validity check is not performed, e.g. as in reoccurring calls to a protected function which does not clear the memory through `clear functions` after a successful license key parse. In order to ensure that the above variables are treated in the same way on every call to the protected function, be sure to clear the memory after the protected function execution has concluded.

For time dependent keys the key generator created by Taurus® during the protection phase looks as follows. It is named after the original file.



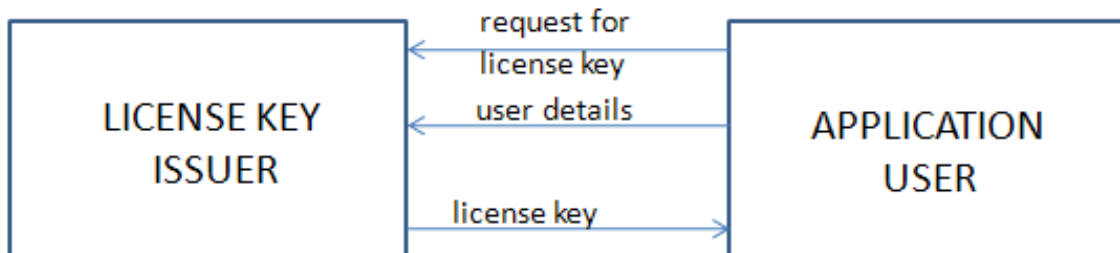
A single license key expiring N days from the date of issuing is generated by clicking “GENERATE KEY(S)”. In the above example, given that the license key is issued on May 30, 2009, the holders of a valid license key file can be configured to see the following license plate every time the application is run in a cleared workspace. Others will not be able to run the application.



You can also create license information plates of your own design, and you can choose to incorporate user information into the license file through the key generator.

3.3 USER DEPENDENT KEY

The procedure for user dependent license key protection is as follows.



A prospective application user issues a request for license along with his/her name and organization. A lifetime license key containing this information is submitted by the license key issuer to him/her.

User dependent license key protection is implemented in Taurus® by choosing "USER DEPENDENT KEY" in the main menu. The below menu then appears.

Taurus license file protection

USER DEPENDENT KEY

source_m_file.m Select m-file

Target file name

Root key

Total number of keys

Scrambler length

Key scrambling rule

License file name

The source m-file is an ordinary function

The source m-file is a GUI created using GUIDE

Use "end" to limit function scope

Disallow program start if license file is corrupted

Disallow program start if license file is not found

Display license information in "LICENSE" menu

Build "REGISTER" menu if license file not ok

Protect m-file and make keygen

powered by algorithmprotection.com

In this menu:

- The file to be protected is selected using "Select m-file" button.
- The target file name is entered in the "Target file name" edit box.
- The root key is a **secret** encryption key consisting of standard ASCII set characters. While there are only few specific requirements for the root key, it should be as "random" a string as possible. **It is equally important that the root key is not distributed to the end users of your application** (otherwise they will potentially be able to bypass the copy protection).
- The total number of keys and scrambler length are not relevant for this protection framework, and cannot be changed.
- License file name corresponds to the file name from which your protected application attempts to read the license key.

- The ordinary function / GUI checkboxes as well as the function scope limiter string checkbox work similarly as in password protection.

The most important features for m-file protection are the “disallow program start if license file is corrupted” and “disallow program start if license file is not found” checkboxes. By checking both of them (default), one can ensure that the application can only be run by any holder of a valid license key. We recommend that you experiment with the functionality of these checkboxes to ensure a proper functionality for your application before releasing it.

If you want your application to display license information in a pre-designed information box, check “Display license information in “LICENSE” menu”. If you want Taurus to implement built-in features for requesting renewal of an expired license etc., check “Build “REGISTER” menu if license not ok”. Note that in this protection framework these features can be built into the application without Taurus®, too.

If you want to use the name of the licensed user in your own code, place the following lines at the beginning of your main function (or opening function if protecting a GUI made using GUIDE):

```
registered_to = '';
try
    eval('registered_to = registered_user;');
catch
end
```

The variable `registered_to` will be assigned the user name encrypted into the license file. If a valid license can be read from a file, this variable is not empty after the above try-catch structure.

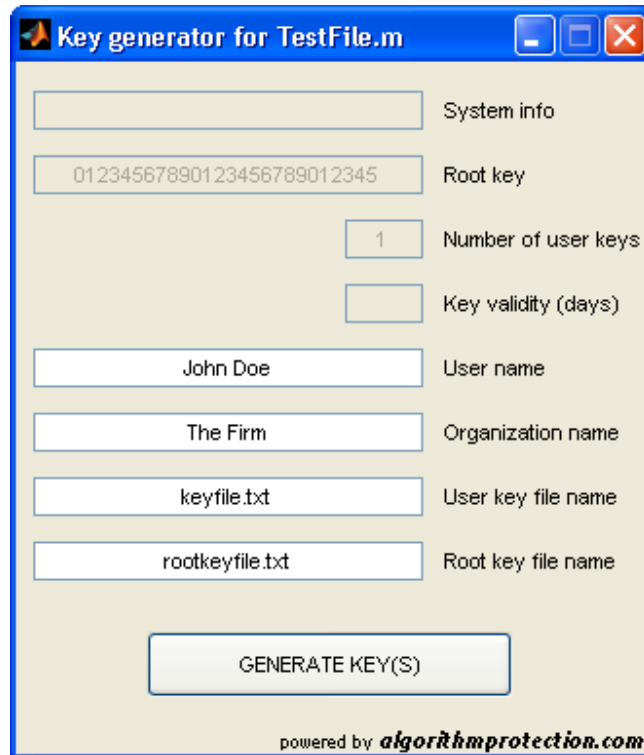
If you want to use the name of the organization of the licensed user in your own code, place the following lines at the beginning of your main function (or opening function if protecting a GUI made using GUIDE):

```
registered_organization = '';
try
    eval('registered_organization = registered_company;');
catch
end
```

The variable `registered_organization` will be assigned the organization name encrypted into the license file. If a valid license can be read from a file, this variable is not empty after the above try-catch structure.

REMARK: The above variables remain empty if a license file validity check is not performed, e.g. as in reoccurring calls to a protected function which does not clear the memory through `clear` functions. In order to ensure that the above variables are treated in the same way on every call to the protected function, be sure to clear the memory after the protected function execution has concluded.

For user dependent keys the key generator created by Taurus® during the protection phase looks as follows. It is named after the original file.



The screenshot shows a Windows-style dialog box titled "Key generator for TestFile.m". It contains several input fields and a button. The fields are: "System info" (empty), "Root key" (containing "01234567890123456789012345"), "Number of user keys" (containing "1"), "Key validity (days)" (empty), "User name" (containing "John Doe"), "Organization name" (containing "The Firm"), "User key file name" (containing "keyfile.txt"), and "Root key file name" (containing "rootkeyfile.txt"). A "GENERATE KEY(S)" button is at the bottom. The footer text reads "powered by algorithmprotection.com".

A single license key (in the above example for "John Doe" of "The Firm") is generated by clicking "GENERATE KEY(S)". In the above example, given that the license key is issued on May 30, 2009, the holders of "license.dat" can be configured to see the following license plate every time the application is run in a cleared workspace. Others will not be able to run the application.

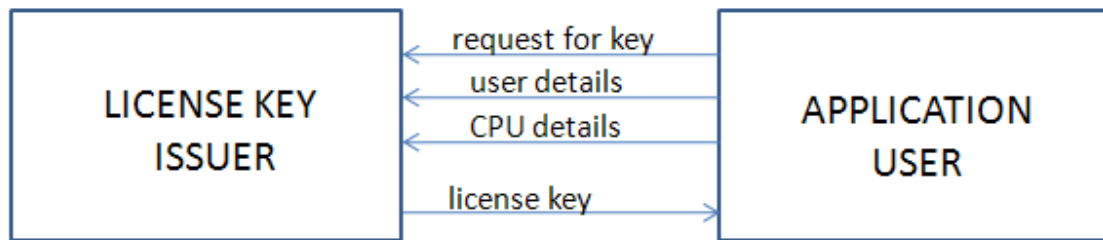


The screenshot shows a Windows-style dialog box titled "License information". It displays the text "This application is licensed to:" followed by "Name: John Doe" and "Organization: The Firm". The footer text reads "powered by algorithmprotection.com".

You can also create license information plates of your own design.

3.4 USER AND COMPUTER DEPENDENT KEY

The procedure for user and computer dependent license key protection is as follows.



A prospective application user issues a request for a license key along with his/her name and organization and details of the computer on which the application is to be run. A lifetime license key containing this information is submitted by the license key issuer to him/her. Subsequently, the application can only be run on the computer specified in the license key, but there is no time constraint in this protection framework.

REMARK: Taurus® implements such routines into the protected application which can automatically gather the relevant information on the user and the computer, and either store this information to a license request file or email it directly to an address of your preference. See Section 4 for details.

User and computer dependent license key protection is implemented in Taurus® by choosing “USER & COMPUTER DEPENDENT KEY” in the main menu. The below menu then appears.

Taurus license file protection

USER & COMPUTER DEPENDENT KEY

source_m_file.m Select m-file

Target file name

Root key

Total number of keys

Scrambler length

Key scrambling rule

License file name

The source m-file is an ordinary function

The source m-file is a GUI created using GUIDE

Use "end" to limit function scope

Disallow program start if license file is corrupted

Disallow program start if license file is not found

Display license information in "LICENSE" menu

Build "REGISTER" menu if license file not ok

Protect m-file and make keygen

powered by algorithmprotection.com

In this menu:

- The file to be protected is selected using "Select m-file" button.
- The target file name is entered in the "Target file name" edit box.
- The root key is a **secret** encryption key consisting of standard ASCII set characters. While there are only few specific requirements for the root key, it should be as "random" a string as possible. **It is equally important that the root key is not distributed to the end users of your application** (otherwise they will potentially be able to bypass the copy protection).
- The total number of keys and scrambler length are not relevant for this protection framework, and cannot be changed.
- License file name corresponds to the file name from which your protected application attempts to read the license key.

- The ordinary function / GUI checkboxes as well as the function scope limiter string checkbox work similarly as in password protection.

The most important features for m-file protection are the “disallow program start if license file is corrupted” and “disallow program start if license file is not found” checkboxes. By checking both of them (default), one can ensure that the application can only be run by any holder of a valid license key. We recommend that you experiment with the functionality of these checkboxes to ensure a proper functionality for your application before releasing it.

If you want your application to display license information in a pre-designed information box, check “Display license information in “LICENSE” menu”. If you want Taurus to implement built-in features for requesting renewal of an expired license etc., check “Build “REGISTER” menu if license not ok”. Since in this protection framework, only a licensed user can run the application on a pre-specified computer, the protected file must have a functionality to gather this information from a prospective user automatically (in the event that he/she does not have a valid license). This functionality is implemented by Taurus® into the protected file, and it is explained in detail in Section 4.

If you want to use the name of the licensed user in your own code, place the following lines at the beginning of your main function (or opening function if protecting a GUI made using GUIDE):

```
registered_to = '';
try
    eval('registered_to = registered_user;');
catch
end
```

The variable `registered_to` will be assigned the user name encrypted into the license file. If a valid license can be read from a file, this variable is not empty after the above try-catch structure.

If you want to use the name of the organization of the licensed user in your own code, place the following lines at the beginning of your main function (or opening function if protecting a GUI made using GUIDE):

```
registered_organization = '';
try
    eval('registered_organization = registered_company;');
catch
end
```

The variable `registered_organization` will be assigned the organization name encrypted into the license file. If a valid license can be read from a file, this variable is not empty after the above try-catch structure.

REMARK: The above variables remain empty if a license file validity check is not performed, e.g. as in reoccurring calls to a protected function which does not clear the memory through `clear functions`

after a successful license key parse. In order to ensure that the above variables are treated in the same way on every call to the protected function, be sure to clear the memory after the protected function execution has concluded.

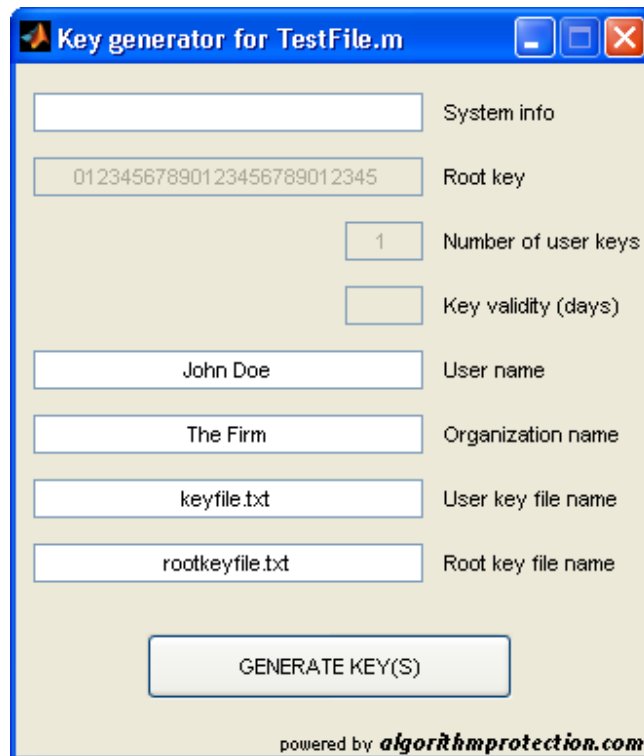
When attempting to run an unregistered version of the application (or when the license key file is not found or is corrupted), the prospective user may be prompted with a dialogue box such as:



It is important that the dialogue box is filled on that computer which is to be used for running the licensed application; the protected application will not execute on other computers with the same license key. After clicking "OK", depending on the user's and code author's preferences, the following information is either saved to a license request text file or sent directly to the license key issuer's email address (see Section 4):

```
Jane Doe <-> A Company <-> Jane.Doe@company.com
58-65-57-60-61-57-58-66-58-65-57-58-64-57-60-58-60-59-60-66-130-66-64-42-80-
107-119-115-118-131-42-64-42-87-121-110-111-118-42-60-60-42-93-126-111-122-
122-115-120-113-42-59-83-120-126-111-118-50-92-51-42-77-111-118-111-124-121-
120-50-92-51-42-77-90-95-42-42-42-42-42-42-42-42-42-63-64-58-42-42-74-42-
60-56-59-61-81-82-132-95-125-111-124-77-58-59-80-55-62-61-59-61-60-59-61-64-
60-59-59-62-63-64-93-55-59-55-63-55-60-59-55-61-67-66-58-63-66-66-60-67-60-
55-61-67-61-64-59-63-59-59-63-67-55-59-58-59-59-62-61-63-67-65-66
```

This information is inputted by the license key issuer into the key generator. For user and computer dependent keys the key generator created by Taurus® during the protection phase looks as follows. It is named after the original file.



In this protection framework, the license key issuer can readily extract the name and organization details of the prospective user from the license request and copy them to the key generator. Moreover, the number sequence should be pasted as such by the license key issuer from the corresponding license request (email or file from the license requestor) to the “System info” edit box in the key generator. In the above example, the following sequence is pasted to the “System info” edit box without line feeds or other special characters:

```
58-65-57-60-61-57-58-66-58-65-57-58-64-57-60-58-60-59-60-66-130-66-64-42-80-
107-119-115-118-131-42-64-42-87-121-110-111-118-42-60-60-42-93-126-111-122-
122-115-120-113-42-59-83-120-126-111-118-50-92-51-42-77-111-118-111-124-121-
120-50-92-51-42-77-90-95-42-42-42-42-42-42-42-42-42-63-64-58-42-42-74-42-
60-56-59-61-81-82-132-95-125-111-124-77-58-59-80-55-62-61-59-61-60-59-61-64-
60-59-59-62-63-64-93-55-59-55-63-55-60-59-55-61-67-66-58-63-66-66-60-67-60-
55-61-67-61-64-59-63-59-59-63-67-55-59-58-59-59-62-61-63-67-65-66
```

A single license key is generated by clicking “GENERATE KEY(S)”. Note that this license key only works on the computer used for generating the license key **request**.

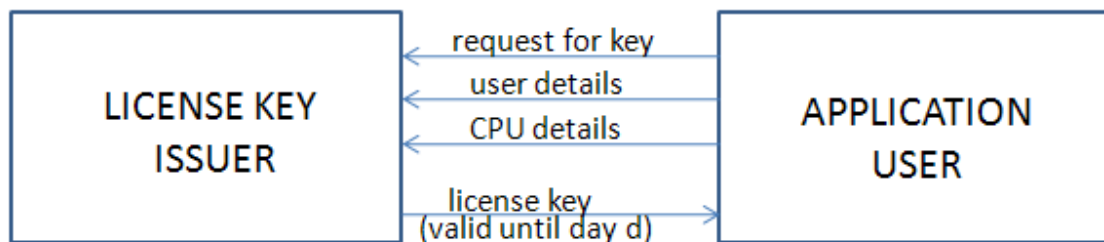
It is possible to display the licensed user in a pre-built license information box, such as:



You can also create license information plates of your own design.

3.5 USER AND COMPUTER AND TIME DEPENDENT KEY

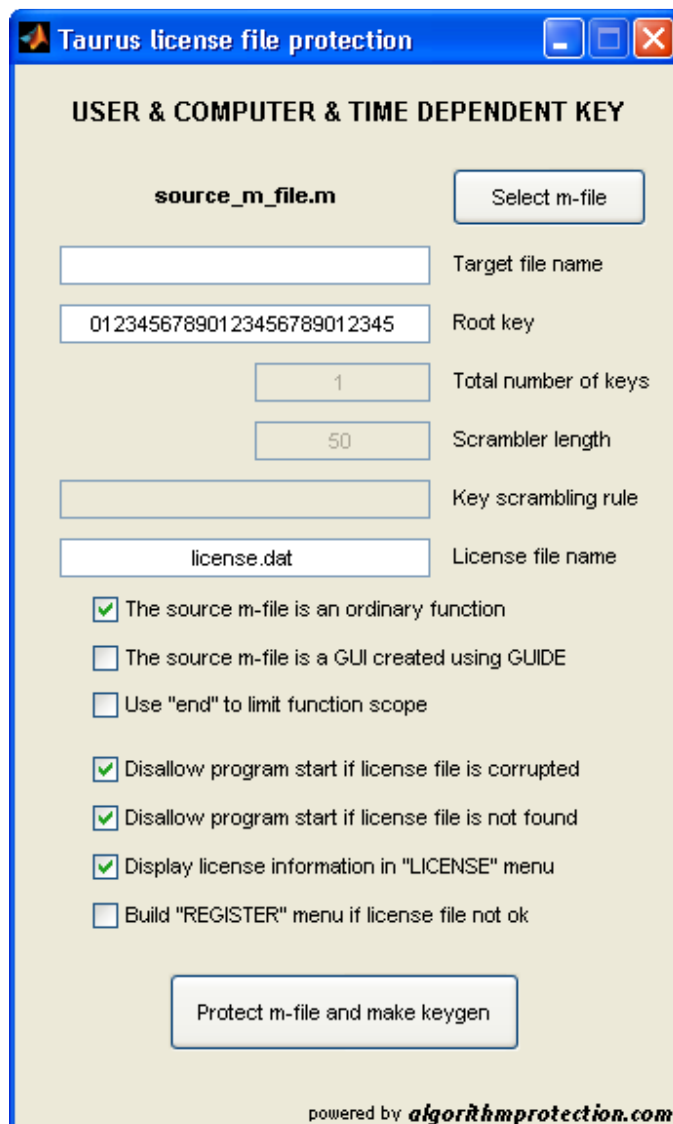
The procedure for user and computer and time dependent license key protection is as follows.



A prospective application user issues a request for a license key along with his/her name and organization and details of the computer on which the application is to be run. A license key containing this information is then submitted by the license key issuer to him/her. This license key is valid until a future date specified by the license key issuer during the key generation process. The protected application can only be run on the computer specified in the license key until this date. The entire process is repeated every time the license key has expired.

REMARK: Taurus® implements such routines into the protected application which can automatically gather the relevant information on the user and the computer, and either store this information to a license request file or email it directly to an address of your preference. See Section 4 for details.

User and computer and time dependent license key protection is implemented in Taurus® by choosing “USER & COMPUTER & TIME DEPENDENT KEY” in the main menu. The below menu then appears.



In this menu:

- The file to be protected is selected using “Select m-file” button.
- The target file name is entered in the “Target file name” edit box.
- The root key is a **secret** encryption key consisting of standard ASCII set characters. While there are only few specific requirements for the root key, it should be as “random” a string as possible. **It is equally important that the root key is not distributed to the end users of your application** (otherwise they will potentially be able to bypass the copy protection).
- The total number of keys and scrambler length are not relevant for this protection framework, and cannot be changed.

- License file name corresponds to the file name from which your protected application attempts to read the license key.
- The ordinary function / GUI checkboxes as well as the function scope limiter string checkbox work similarly as in password protection.

The most important features for m-file protection are the “disallow program start if license file is corrupted” and “disallow program start if license file is not found” checkboxes. By checking both of them (default), one can ensure that the application can only be run by any holder of a valid license key. We recommend that you experiment with the functionality of these checkboxes to ensure a proper functionality for your application before releasing it.

If you want your application to display license information in a pre-designed information box, check “Display license information in “LICENSE” menu”. If you want Taurus® to implement built-in features for requesting renewal of an expired license etc., check “Build “REGISTER” menu if license not ok”. Since in this protection framework, only a licensed user can run the application on a pre-specified computer, the protected file must have a functionality to gather this information from a prospective user automatically (in the event that he/she does not have a valid license). This functionality is implemented by Taurus® into the protected file, and it is explained in detail in Section 4.

If you want to use information on the number of days until license expiry in your own code, place the following lines at the beginning of your main function (or opening function if protecting a GUI made using GUIDE):

```
license_days_left = 0;
try
    eval('license_days_left = free_trial_days_left;');
catch
end
```

If you want to use the name of the licensed user in your own code, place the following lines at the beginning of your main function (or opening function if protecting a GUI made using GUIDE):

```
registered_to = '';
try
    eval('registered_to = registered_user;');
catch
end
```

The variable `registered_to` will be assigned the user name encrypted into the license file. If a valid license can be read from a file, this variable is not empty after the above try-catch structure.

If you want to use the name of the organization of the licensed user in your own code, place the following lines at the beginning of your main function (or opening function if protecting a GUI made using GUIDE):

```

registered_organization = '';
try
    eval('registered_organization = registered_company;');
catch
end

```

The variable `registered_organization` will be assigned the organization name encrypted into the license file. If a valid license can be read from a file, this variable is not empty after the above try-catch structure.

REMARK: The above variables remain empty and `license_days_left` remains 0 if a license file validity check is not performed, e.g. as in reoccurring calls to a protected function which does not clear the memory through `clear` functions after a successful license key parse. In order to ensure that the above variables are treated in the same way on every call to the protected function, be sure to clear the memory after the protected function execution has concluded.

When attempting to run an unregistered version of the application (or when the license key file is not found or is corrupted), the prospective user may be prompted with a user dialogue box such as:



It is important that the dialogue box is filled on that computer which is to be used for running the licensed application; the protected application will not execute on other computers with the same license key. After clicking "OK", depending on the user's and code author's preferences, the following information is either saved to a license request text file or sent directly to the license key issuer's email address (see Section 4):

```

Jane Doe <-> A Company <-> Jane.Doe@company.com
58-65-57-60-61-57-58-66-58-65-57-58-64-57-60-58-60-59-60-66-130-66-64-42-80-
107-119-115-118-131-42-64-42-87-121-110-111-118-42-60-60-42-93-126-111-122-
122-115-120-113-42-59-83-120-126-111-118-50-92-51-42-77-111-118-111-124-121-
120-50-92-51-42-77-90-95-42-42-42-42-42-42-42-42-42-63-64-58-42-42-74-42-
60-56-59-61-81-82-132-95-125-111-124-77-58-59-80-55-62-61-59-61-60-59-61-64-
60-59-59-62-63-64-93-55-59-55-63-55-60-59-55-61-67-66-58-63-66-66-60-67-60-
55-61-67-61-64-59-63-59-59-63-67-55-59-58-59-59-62-61-63-67-65-66

```

This information is inputted by the license key issuer into the key generator. For user and computer and time dependent keys the key generator created by Taurus during the protection phase looks as follows. It is named after the original file.

Key generator for TestFile.m

58-65-57-60-61-57-58-66-58-65-57-58-E System info

01234567890123456789012345 Root key

1 Number of user keys

365 Key validity (days)

Jane Doe User name

A Company Organization name

license.dat User key file name

rootkeyfile.txt Root key file name

GENERATE KEY(S)

powered by algorithmprotection.com

In this protection framework, the license key issuer can readily extract the name and organization details of the prospective user from the license request and copy them to the key generator. Moreover, the number sequence should be pasted as such by the license key issuer from the corresponding license request (email or file from the license requestor) to the “System info” edit box in the key generator. In the above example, the following sequence is pasted to the “System info” edit box without line feeds or other special characters:

```
58-65-57-60-61-57-58-66-58-65-57-58-64-57-60-58-60-59-60-66-130-66-64-42-80-
107-119-115-118-131-42-64-42-87-121-110-111-118-42-60-60-42-93-126-111-122-
122-115-120-113-42-59-83-120-126-111-118-50-92-51-42-77-111-118-111-124-121-
120-50-92-51-42-77-90-95-42-42-42-42-42-42-42-42-42-42-63-64-58-42-42-74-42-
60-56-59-61-81-82-132-95-125-111-124-77-58-59-80-55-62-61-59-61-60-59-61-64-
60-59-59-62-63-64-93-55-59-55-63-55-60-59-55-61-67-66-58-63-66-66-60-67-60-
55-61-67-61-64-59-63-59-59-63-67-55-59-58-59-59-62-61-63-67-65-66
```

The number of days the license key is valid is entered into the “Key validity (days)” edit box.

A single license key is generated by clicking “GENERATE KEY(S)”. Note that this license key only works on the computer used for generating the license key **request**, and only for the number of days specified in the license key.

It is possible to display the licensed user in a pre-built license information box, such as:



You can also create license information plates of your own design.

4 REGISTER PRODUCT MENU

Taurus® is able to implement graphical popup windows to notify the user of the protected software for a missing, corrupted or expired license key, and provide a menu for requesting a working license key. You can enable this feature in your application by checking “Build “REGISTER” menu if license not ok” in Taurus® after selecting an appropriate license key protection scheme. Once you check this checkbox, you are asked to fill in information on the auto-request email account:

License auto-request

Auto-sender email address:
matlab.automailer.test@gmail.com

Auto-sender email address password:
jeNBienae37fOvQA

Auto-sender SMTP server:
smtp.gmail.com

Auto-sender SMTP mail server authentication:
true

Auto-sender SMTP socket factory class:
javax.net.ssl.SSLSocketFactory

Auto-sender SMTP socket factory port:
465

License administrator email address:
your_email@your_address.com

License auto-request message subject line:
License request

OK Cancel

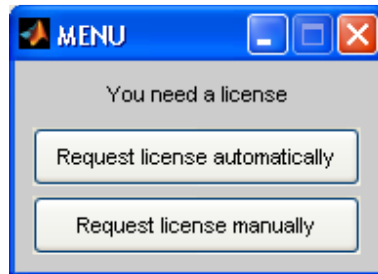
The sample answers displayed as default ones use Google's Gmail® because of its simplicity, and the text to follow assumes a Gmail account for automatic registration. However, you are free to utilize other SMTP mail systems, too, with appropriate modifications.

Once you have generated a Gmail account for the automatic mailer, you only need to set its email address to the "Auto-sender email address" field and the corresponding Gmail login password to "Auto-sender email address password" field. The email address of the license key issuer (license key administrator) should be entered to the "License administrator email address" to replace "your_email@your_address.com". If you want to have a specific message subject string, you should place that string to the last field on the dialogue box above. Click OK to confirm your choices.

REMARK: Apart from the license issuer (administrator) email address, none of the information entered into the license auto-request configurator is displayed to the user of the application.

Note that the above sample account "matlab.automailer.test@gmail.com" and the password are valid (as of 6/2009) and can be used for testing purposes. In this case you should just put your email address to the "License administrator email address" field to begin receiving license key requests. Bear in mind that this is a public test account, so you should avoid using sensitive email addresses etc. in this context.

Once your application is protected with “Build “REGISTER” menu if license not ok” checked, a user without a proper license key file has the two alternatives displayed in the below menu box upon application execution:



- REQUEST LICENSE AUTOMATICALLY: The user is prompted for certain information in a question dialogue box, and the autosender (see above) email account is used for sending this information directly to the license key issuer (administrator).
- REQUEST LICENSE MANUALLY: The user is prompted for certain information in a dialogue box, and the answers are saved to a license request file. The user is informed that this file should be sent to the license key issuer (administrator) at the address given during protection stage.

In both cases, the information inputted by the license requestor is name, organization and email address:

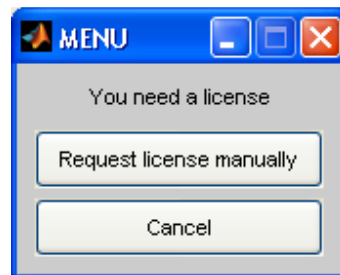


The information written to a file or sent to the license key issuer (administrator) always has the same format:

```
Jane Doe <-> A Company <-> Jane.Doe@company.com
58-65-57-60-61-57-58-66-58-65-57-58-64-57-60-58-60-59-60-66-130-66-64-42-80-
107-119-115-118-131-42-64-42-87-121-110-111-118-42-60-60-42-93-126-111-122-
122-115-120-113-42-59-83-120-126-111-118-50-92-51-42-77-111-118-111-124-121-
120-50-92-51-42-77-90-95-42-42-42-42-42-42-42-42-42-63-64-58-42-42-74-42-
60-56-59-61-81-82-132-95-125-111-124-77-58-59-80-55-62-61-59-61-60-59-61-64-
60-59-59-62-63-64-93-55-59-55-63-55-60-59-55-61-67-66-58-63-66-66-60-67-60-
55-61-67-61-64-59-63-59-59-63-67-55-59-58-59-59-62-61-63-67-65-66
```

The name and organization information are to be entered into the key generator. The number string corresponding to computer information is also entered to the key generator if the application employs a computer specific protection framework. The license key generated by the key generator is to be sent to the email address displayed in the message.

REMARK: An issue arises if your application is protected using one of the two computer dependent license key frameworks, without having “Build “REGISTER” menu if license file not ok” checked. In this case there still has to be a way for an unlicensed user to request a license key corresponding to his/her computer. Taurus® builds this functionality conveniently into your m-files. In this case, the user is prompted with the following menu:



The functionality of the manual license request button is the same as explained above. However, in this case there is no license issuer (administrator) address visible for the user; this information has to be communicated by other means.

5 FILE ENCRYPTION

TaurusEncrypt® and TaurusEncrypt-R® are the file encryption tools, which are only included in the Taurus® PROFESSIONAL distribution package. You can use them to:

- Obfuscate source code (if distributed along with your application) from unlicensed users;
- Prevent unauthorized use of critical data files.

TaurusEncrypt® and TaurusEncrypt-R® use the Taurus® license key format strings as keys to encrypt files. This means that you have the possibility – but not an obligation – to encrypt your data using the same license key(s) as your copy protected application uses to check the user’s identity etc. In this case, you can deliberately allow the licensed user to be able to decrypt the data files and/or source code with the license key that was delivered to him/her. Of course, you can keep the file encryption key secret, in which case the process works as follows:

- You encrypt the data files with TaurusEncrypt® prior to distributing your application;
- You include TaurusEncrypt-R® along with your application distribution package (your Taurus® PROFESSIONAL license guarantees that this can be done free of charge and royalties);

- Your application calls TaurusEncrypt-R® during its running – TaurusEncrypt-R® performs on-the-fly file decryption rapidly in the system memory using the secret key delivered to it as a parameter (not read from file). No plaintext is ever written to a file by TaurusEncrypt-R®.
- Your application handles processing of the decrypted data.

Refer to the sample Taurus® code package for an example of the above. The “README.txt” file found in the “File_encryption” folder of the Taurus® sample code package also includes a hands-on example to see TaurusEncrypt® and TaurusEncrypt-R® in action.

6 GENERAL GUIDELINES

In this section we list some general guidelines for successfully protecting your MATLAB® code with Taurus.

- The sample case package available for downloading at www.algorithmprotection.com contains hands-on information on how to implement a free trial version of a MATLAB® application using Taurus.
- Some Taurus® functionality relies on Java, which is included with MATLAB®. Taurus® has not been tested in the `-nojvm` mode, so be sure to run your application with Java on.
- When protecting graphical user interfaces created with GUIDE, the first line of the opening function should end with `_OpeningFcn(hObject, eventdata, handles, varargin)`. In particular, you should use the argument variable names generated by GUIDE.
- Since Taurus® only protects one m-file at a time, you should strive to include all your critical functions in that file. Alternatively, you can deploy the application to a stand-alone executable in which case protecting the main function with Taurus® suffices.
- Taurus® operates on the front-end of your algorithms (function files). Therefore, it is advisable to have the opening of the main function as slim as possible, containing at most an input variable processor, e.g.:

```
function Out = StartupFunction(varargin)
if nargin<1
    input=NaN;
else
    [[[ PUT SOME INPUT VARIABLE PROCESSING HERE ]]];
end
Out = TheFunctionIReallyWantToRun(input);
```

In this example “TheFunctionIReallyWantToRun” is the (main) function to be protected. The above construction ensures that Taurus® will not interfere with the initialization of your main function.

- Taurus® does not directly protect c code intended to be compiled to .mex files within MATLAB®. If your application uses proprietary .mex files, one alternative for using Taurus® to protect it is as follows: Create a main function (m-file) which invokes calls to the .mex files, protect it with Taurus® and finally compile the set of files to a stand-alone application.
- If your application consists of two or more interlinked modules created using GUIDE, with separate .fig files in each, you should consider protecting them all individually with Taurus®. In this case the best choice is to use precisely the same protection settings (root key) on each m-file corresponding to a .fig file, because then you need only one license key file for the entire application (its presence and validity are verified by all of the modules independently of each other). Note, however, that the .fig files are not encrypted. If this is needed, you should compile the application to a stand-alone one.
- It may not be possible to directly compile the p-files generated by Taurus®, but you can circumvent this by creating a simple m-file front-end around it, e.g. as follows:

```
function Outputs = MySimpleFrontEnd(varargin)
Inputs = varargin;
Outputs = MyPCodedFunction(Inputs);
```

Here MyPCodedFunction is the file created by Taurus®, and the MySimpleFrontEnd (m-file) can be compiled seamlessly on MATLAB Compiler 4.0 (R14) and later; refer to the information contained in <http://www.mathworks.com/support/solutions/data/1-1BZEA.html>.

7 PRACTICAL EXAMPLES OF CONTROLLED LICENSE KEY DISTRIBUTION

In this section we briefly describe how to achieve secure and efficient license key distribution in two examples. The first example utilizes the USER INDEPENDENT LICENSE KEY check framework; it is the simplest of the license key protection frameworks supported by Taurus®. The second example utilizes the USER AND COMPUTER AND TIME dependent license key check framework, which provides the strongest copy protection for your application. The rest of the license key protection frameworks supported by Taurus® lay in between these two extremes in terms of information security and complexity; you should decide on a case by case basis which suits your application best.

7.1 SELLING SOFTWARE ONLINE

The USER INDEPENDENT LICENSE KEY check framework is a very potential alternative for copy protecting applications intended to be sold online, especially those on the lower end of cost scale. Its main benefit is that the license key distribution to purchasers of the software can be completely automated through modern inexpensive e-commerce solutions. This implies less time spent on interacting (unproductively)

with clients, allowing more time to be spent on true customer service, development efforts, marketing etc.

Probably the simplest way to succeed is this:

- Build your application in such a way that it embodies a “free trial mode” (see Section 3.1) and only launches in the “full version” mode if a valid license key file is found.
- Protect the application in the USER INDEPENDENT LICENSE KEY check framework provided by Taurus®, and make sure that the free trial works as expected.
- Generate all license keys at once using the key generator built by Taurus®. You can store all license keys as the rows of one master file, say “keys.txt”. You can also duplicate license keys by copying and pasting in the file; just make sure there are enough of valid keys in the file.
- Put the (free trial) software on your web page for free downloading. Make sure that the key generator, the license keys and the root key are **not** placed there.
- Arrange the license key distribution from the master key file through an Online Payment Processor (OPP) such as e-Junkie (no affiliation to algorithmprotection.com). Essentially this means that you place customized “buy now” type buttons on your web page, which the OPP monitors. Once a buy transaction is completed, the OPP sends the next license key in the master file directly to the purchaser.

7.2 COPING WITH HEAVY INFORMATION SECURITY NEEDS

In many industries, especially homeland security and defense technology, it is of paramount importance that certain algorithms and applications are not distributed to or used by unauthorized persons. The USER AND COMPUTER AND TIME DEPENDENT license key check framework of Taurus® is a very potential alternative for copy protecting such applications. Its main benefit is that the license key issuer has full control over the persons running the applications as well as the computers on which this is being done. The license key issuer can also be sure that the critical application will not start after the license has expired, which serves the purpose of “delete after use” order (which is all too often ignored by software users).

Assuming that there is a group of people starting to work on a highly classified project, one way to successfully implement the above strict information security framework is:

- At the start of the project, the Project Owner should nominate a license key issuer (most likely the Project Manager). He/she shall:
 - Decide upon the root key sequence(s) to be used in encryption – one per protected application.
 - Decide upon individual privileges – which applications a given group member can use
 - Protect the applications using the above designed root keys in Taurus®
 - Distribute the applications to the group members

- Be responsible for overall information security in the project, i.e. securely storing the root key(s) and key generators etc.
- Once a group member requires access to an application, he requests a license key from the license key issuer, and possibly includes an estimate of duration of use.
- Assuming the request is a valid one, the license key issuer generates a license key to the requestor. The license key issuer chooses the license key expiry date(s) based on project milestone dates and possibly other information contained in the license key issuer's request.
- Once the project is complete, the license key issuer permanently deletes the key generators and the root keys, and instructs the group members to delete the critical applications (and keys).