KEYLOK®

Software Piracy Prevention System

User Manual

FORTRESS

Copyright

Trademarks

KEYLOK owns a number of registered and unregistered Trademarks and Service Marks (the “Marks”). These Marks are extremely valuable to KEYLOK and shall not be used by you, or any other person, without KEYLOK's express written permission. The Marks include but are not necessarily limited to the following: KEYLOK; BIT-LOCK; S-LOK™; COMPU-LOCK™; and the KEYLOK logo.

S-LOK™ is a joint trademark of KEYLOK and A.S.M. Inc.

You shall not use any of the Trademarks or Service Marks of KEYLOK without the express written permission of such Trademark or Service Mark owner.

KEYLOK

777 S Wadsworth Blvd.
Bldg. 4-220
Lakewood, CO 80226
USA

Phone: (303) 801-0338 or 1-800-4KEYLOK (1-800-453-9365)
Fax: (303) 228-0285 or (720) 294-0275
Email: info@keylok.com
Web: www.keylok.com

Last updated: February 8, 2022

Contents

Windows Quick Start

Follow these steps to protect your Windows application with a KEYLOK Fortress dongle in 30 minutes:

1. **Install the software and run the demo tool.** **5 Minutes**

   a) Using the link, download the .iso disk image file.

   (This link is for the evaluation SDK. You will be sent another with your company unique SDK.)

   b) In the top-level directory run Setup.exe. This will install utilities for the SDK under \Program Files\KEYLOK.

   c) Go to \Program Files\KEYLOK\Demos and run *SDKDemo.exe* to familiarize yourself with the features and functionality of the Fortress Security System.

2. **Compile and run the sample code for your development language.** **5 Minutes**

   Examine the sample source code for your development language and environment to learn how to implement the KEYLOK API calls.

   (The appropriate sample code can be found in the \Documents\KEYLOK\SampleCode\ directory.)

3. **Protect your application.** **20 Minutes**

   Simply copy and paste the desired features from the sample code into your application.

Networking Quick Start

Perform the Windows Quick Start first, and then follow these steps to set up your dongle server and clients so that you can access a centrally mounted dongle over a TCP/IP network:

1. **Set up your dongle server.**

   Run *INSTALL EXE*, (located in the "Documents\KEYLOK\Send To End Users folder) and select the appropriate type of dongle (Fortress) and the "Server" radio button. Click OK to install the server software.

2. **Set up your clients.**

   On each "client" computer that requires access to a dongle mounted on a remote computer, run *INSTALL EXE,* and select the "Client" radio button. Click OK to install the client software.

3. **Familiarize yourself with the network demo software.**

   Run *NetDemo exe* or *VerifyNetworkKey exe* (found in "Program Files\KEYLOK\Networking) on one of your client machines to verify the network connection to the dongle server (verifynetworkkey exe) and familiarize yourself with the network demo applications (netdemo exe). Run *NetKeyMonitor exe* to see which servers are running and how many active dongle sessions each one has.

4. **Set up your application code for networking.**

   Compile your application and link it with the appropriate network-enabled interface file (in the sample code for your development language and environment): nwkl2_32.dll or kfunc32MTn.lib.

   See the ***Dongle Access Over a Network*** section of this manual to gain a full understanding of implementing the network dongle.

NOTE: Any machine in your network that can be pinged by all client machines can be used as a dongle server. The dongle server does not have to run a server operating system.

Commented [RB1]: Nwkl2_32.dll referencing kl2 in fortress manual? Or do we not care?

Evaluation to Production Quick Start

Follow these steps to set up your protected application to use your company-unique production dongles instead of the demo dongle:

1. **Find the demo codes in your application source code. They may be in an included file or embedded directly in the source code.**

   - C/C++: *client h*
   - VB: *GLOBAL BAS*

   For other languages, look for a series of eleven lines of codes starting with *ValidateCode1*.

   Run the Setup.exe located in your company unique SDK to update the Keylok utilities.

   Find your company-unique codes on the company-unique disc enclosed with your first production order.

2. The company-unique files are named *client *.* We provide the files in several different formats. If none of the client files is the right format for your language and development environment, the *YOURCODES DAT* file has the same information in very generic format so that you can edit the "demo" dongle codes and replace them with the codes assigned uniquely to your company.
   Copy your company-unique codes into your application source code in place of the demo codes and recompile.

3. Simply cut and paste your company-unique codes from the file you located in Step 2 in place of the demo codes you found in Step 1. Then recompile and re-link your application and it is ready to use with your production dongles.

NOTE: When switching from demo to production USB dongles, you must:

1. Stop all programs that may be communicating with the dongle,
2. Remove the demo dongle, and
3. Connect the company unique dongle

Mac Quick Start

Follow these steps to protect your Mac OS X gcc application with a USB KEYLOK Fortress in 30 minutes:

1. **Install the software and the dongle.** **5 Minutes**

   Go to folder called: \SampleCode\Mac. The Demo version of the SDK can be found at this link. Drag this folder to your desktop. Attach your dongle to a USB port.

2. **Compile and run the sample code in gcc.** **5 Minutes**

   a) Open Terminal

   b) Go to the Mac folder on the Desktop

   c) Compile the demo:

   make –f Makefile

   d) Run the demo:

   /demoA

3. **Protect your application** **20 Minutes**

   Simply copy and paste the desired features from the sample code into your application.

Linux Quick Start

Follow these steps to protect your Linux application with a USB KEYLOK Fortress dongle in 30 minutes:

1. **Install the software.** **5 Minutes**

   Go to the directory called: \SampleCode\Linux. The evaluation SDK can be found at this link. Copy this directory to a convenient directory.

2. **Compile and run the sample code in gcc.** **5 Minutes**

   a) Attach your demo dongle

   b) Go to the appropriate directory (USB)

   c) Compile the demo:

   make –f makefile

   d) Run the demo:

   /demoA

3. **Protect your application** **20 Minutes**

   Simply copy and paste the desired features from the sample code into your application.

   See the ***Using KEYLOK Under Linux*** section of this manual for more info.

NOTE: Follow the same steps for QNX or FreeBSD but use the sample code and object(s) in the appropriate directory in the sample code.

See the Using KEYLOK Under Linux section for more information.

Commented [RB2]: KL2 Manual says: d) Run the demo: ./demoA for statically linked example or .\demoSO for dynamically linked example.

Should this be the same?

Introduction

KEYLOK is proud to offer the most complete solution to software security in the marketplace. KEYLOK pioneered dongle-based software piracy prevention in 1980 and has continued to lead the industry throughout the history of software protection.

Our product offerings include:

KEYLOK2: The KEYLOK security system provides a very high degree of security with an economical hardware key. KEYLOK dongles are only available ~~for~~ in USB ~~and serial ports~~ on computers running ~~DOS~~, WINDOWS 3.x / 9x / ME / NT / 2000 / XP / Vista / 7 / 8 / 10 / 11/ Server2003 / Server2008 / Server 2008 R2 / Server 2012, LINUX, FreeBSD, QNX or Macintosh OS X. ~~Serial port dongles can be used on any computer with an RS232 port and do not require a device driver.~~ KEYLOK dongles have programmable memory, remote update, lease control algorithms, and networking capability for controlling multiple users with a single hardware lock.

Commented [RB3]: I was wondering if the descriptions of the other products were relevant.

KEYLOK3: KEYLOK3 is backwards compatible with the KEYLOK2 features, provides enhanced security over KEYLOK2, and offers the additional convenience of not requiring product unique device drivers, and is currently available for Windows platforms. The KEYLOK security system provides a very high degree of security with an economical hardware key. KEYLOK3 dongles are available for USB ports on computers running WINDOWS 9x / ME / NT / 2000 / XP / Vista / 7 / 8 / 10 / 11 / Server2003 / Server2008 / Server 2008 R2 / Server 2012. KEYLOK dongles have programmable memory, remote update, lease control algorithms, and networking capability for controlling multiple users with a single hardware lock.

FORTRESS: Backwards compatible with KEYLOK2, the dongle also offers two distinct differences. Firstly, the dongle operates in HID mode: you do NOT need to install a device driver. And secondly, Fortress allows you to migrate functions from your code to execute only on the dongle providing you with an unparalleled security solution. Fortress also provides larger user memory (5K - 51K bytes) and provides an added level of tamper resistance, increasing the protection against piracy through reverse engineering.

Feature Comparison

Feature	KEYLOK2	KEYLOK3	Fortress
Hardware Type	USB	USB	USB
Driverless		**ü**	**ü**
User Memory (Read/Write) bytes	112	112	5k-55k
Expiration Date	**ü**	**ü**	**ü**
Remote Update	**ü**	**ü**	**ü**
Counters	**ü**	**ü**	**ü**
Anti-Debugger	**ü**	**ü**	**ü**
Network Access	**ü**	**ü**	**ü**
Smart Card			**ü**
Tamper Resistant			**ü**
Code Vault (Optional)			**ü**
Flash Drive (1-16GB Optional)			**ü**
Real Time Clock (Optional)			**ü**

General Product Overview

The KEYLOK security system protects your software applications from piracy, thereby increasing your revenues associated with software sales. The security is transparent to your end-user once the hardware dongle is installed in the computer's USB port. Unlimited backup copies of the program can be made to protect your clients with the knowledge that for each copy of your software sold only one copy can be in use at any one time. Your clients can install the software on multiple machines (~~e.g.~~e.g., at the office and at home) without having to go through difficult and time-consuming install/uninstall operations. Your clients can easily restore or reinstall copies of your software following catastrophic events such as hard disk failures. These advantages provide your clients with the features they desire and deserve while preserving your financial interests.

The KEYLOK security system uses ~~a number of~~several sophisticated techniques for verification of hardware dongle presence. The KEYLOK is also provided with 112 bytes (5K -51K bytes for Fortress) of field programmable read/write memory. There are NO special programming adapters required to program the dongle memory.

When first attempting to communicate with the hardware dongle it is necessary to successfully complete an exchange of bytes between the host and the dongle that differs during each dongle use (i.e., using an active algorithm). If this sequence is properly executed the dongle will return a customer unique 32-bit identification code which you can use as confirmation that one of your hardware security dongles is installed on the computer. If an improper data exchange ~~occurs~~occurs, then the security system returns to the host a random 32-bit code in place of the proper identification code. Upon successful completion of the authentication sequence, the host computer then sends a 48-bit customer unique password to the dongle to authorize memory read operations. Memory write operations require the successful transmission of yet another 48-bit customer unique password to the dongle. The write password must be sent AFTER transmission of the proper READ authorization sequence. If the dongle is sent the incorrect read/write ~~password~~password, then subsequent memory operations are ~~ignored~~ignored, and random information is returned to the program. In summary, a total of 176 bits of customer unique codes must be known ~~in order to~~to alter the memory within the dongle.

Up to fifty-six (56) separate 16-bit counters (values of 0-65535) can be independently maintained within the dongle. Counters are particularly useful for controlling demonstration copies of software, as well as pay-per-use software (~~e.g.~~e.g., testing). Some clients use the counter as a means of controlling software use up until the time they have been paid for the software, and then provide their clients with a special code that 'unlocks' the dongle for unlimited future use.

At the time of manufacturing each dongle is programmed with a unique dongle serial number, thus providing you with the capability of identifying the specific dongle (and thus specific end-user) for customers requiring this level of control.

The security system includes algorithms for performing very secure remote memory modifications to the dongle. The remote update procedure involves the exchange back and forth between the end-user and you, of a series of numeric values displayed on the system console. These values are entered into the security system to activate a desired memory change at the end-user's site. This can be used to query memory, replace memory contents, extend counters, extend lease expiration dates, add additional network licenses, etc. The specific sequence used to ~~effect~~affect a memory change will only work one time, and only on one specific dongle. Various solutions for remote update have been provided. You should select the one most appropriate and consistent with the type of interface you expect to have with your end users. See the Remote Update section of this manual for a more thorough explanation of the available options.

Sophisticated algorithms allow the client's system clock to be used as an economical means of controlling leased software. The most recent system clock date and time are stored internally within the KEYLOK memory. Any attempt by the end-user to set back the date and/or time generates appropriate error codes to your application. Depending upon your needs, a real time clock option is available for the Fortress LS dongle.

The KEYLOK price-to-feature ratio is unparalleled in the industry.

Implementation Overview

KEYLOK protection for your application is implemented by embedding calls to our Application Programming Interface (API) into your application source code (see Fig. 1). The references to our API are satisfied by linking your code to either a Windows DLL or a library file (depending upon your programming language and development environment).

In most cases, the bulk of the necessary modifications to your source code can be made simply by cutting and pasting from the sample code supplied with our Software Development Kit (SDK). You can choose to implement any level of security you wish, depending only on which features of KEYLOK you want to use. The simplest and most basic security, a simple check for the presence of the KEYLOK, can be implemented in fewer than 20 lines of code.

If you need to pre-program your dongles (to set a lease expiration date or a usage counter, for example), you can do so using KLTOOL.exe that we supply. We also supply utilities for verifying dongle installation is working correctly (verifykey.exe/verifynetworkkey.exe) and for remotely updating the contents of the dongle memory or the lease expiration date by sending the end user a secured file.

Networking (in which one dongle is mounted on a central dongle server and accessed remotely by clients via a TCP/IP network) is equally easy to implement. You only need to link your application with the network version of our DLL or object file and install the appropriate service. Our TCP/IP network dongles are available in versions that support from 3 to an unlimited number of simultaneous users (our standard dongles will support one user at a time over a network connection). We supply utilities for verifying that a remote dongle can be accessed and for diagnosing common problems with the dongle access via a network.

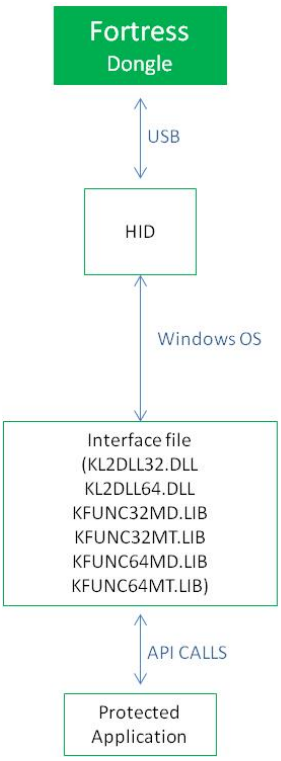

Figure 1: How Dongles Communicate with a Protected Application

Each KEYLOK dongle is individually programmed by KEYLOK before shipping with a set of access codes and a serial number, which ~~can't~~cannot be changed after programming. When you first purchase production dongles, we assign your company a unique set of codes that will allow you and only you to access your dongles. We maintain an extensive set of internal security procedures to protect your company-unique access codes and to ensure that they are never sent to anyone except your designated contacts without your prior written authorization. We will provide you with these codes ~~on a disc~~ in a company unique SDK that will be included with your first production order.

The demo dongles that are included in our evaluation kits are identical in every way to our production dongles, except that they are all programmed with company-unique information for the same fictitious company, and the access codes necessary to communicate with the demo dongles are built into ~~all of~~all our sample code. To convert from using a demo dongle to using your production dongles, all you need to do is cut and paste the codes we send you in place of the demo access codes and recompile your application. We also supply company-unique versions of our dongle programming and troubleshooting utilities.

Getting Started

Protecting your application can be done in 3 simple steps:

Step 1: Install the Software Development Kit

The KEYLOK Software Development Kit (SDK) contains all the necessary hardware and software to protect your application.

The installation program has an easy-to-use graphical interface. ~~Insert the KEYLOK SDK CD into the CD-ROM drive. The CD will automatically run under Windows. If the CD does not automatically run, execute d:*setup.exe* manually. ('d' is the directory of the CD-ROM drive).~~ To access the KEYLOK SDK, download the '.iso' file provided by our development team. To get started, run the setup.exe which is located in the root directory inside the '.iso' file.

Formatted: Not Highlight

Step 2: Run the sample source code

Sample code is provided for over 50 different compilers and development tools. If you did not see sample code for your compiler during ~~installation~~installation, please contact technical support.

Examine the source code for the demonstration program (DEMO.xxx) written in the software development language that you are using to see how to implement the KEYLOK API calls. Sample source code will be installed in the following directory:

~~Documents\KEYLOK\SampleCode\~~ Documents\KEYLOK\Sample Code\<operating system>\<language>

Application specific notes are provided in the form of a *README.TXT* file. Sample code for older programming languages and development environments can be found in:

~~Documents\KEYLOK\Samplecode\~~ Documents\KEYLOK\Sample Code\Archive\<language>

Step 3: Move relevant portions of the sample code to your application

Once you are familiar with the KEYLOK API calls simply copy and paste the relevant calls into your application.

Review the ***Security Considerations*** section of this manual for ideas on how to increase the security ~~using the KEYLOK system.~~ of your application through utilizing KEYLOK dongles.

NOTE: Access to the KEYLOK API calls is accomplished through a LIB file or DLL file depending on your development tool. The appropriate interface file will be found in the sample code directory for your programming language and development environment.

Security Considerations

The following suggestions are intended to help you increase the level of protection built into your application.

Enhancing Security

The ***Check for KEYLOK*** process in the demonstration program involves a comparison of actual return values to expected values. This technique is open to debugger analysis and thus being patched around by experienced software pirates. A much more secure approach is to use the returned values in calculations performed in your program, and only much later in the program to analyze the results of the calculation to determine whether proceed with program execution or to exit. The more you can bury the decision process, by which you conclude the presence of a proper security dongle, the more effective the total security system will be.

Use of Dongle Memory

Memory within the security dongle can be used for many purposes, as follows:

- The most common use of dongle memory is to control licensing of multiple ~~product~~products and/or features within a product. You have the option of using a complete word (16 bits) of memory to control a single feature or alternately to use individual bits within a single memory address to perform the same task.
- Product revision control information can be stored within dongle memory. See the ***Product Version Control*** section below that provides additional suggestions regarding this capability.
- Store client name and or copyright message as text within the dongle for display from within your application.
- Store critical constants used in calculations within your program.
- Use memory for counter(s) to control client billing whereby charges are made for each service performed by the program, rather than a fixed purchase price for the software.

A random word can be written to the dongle during one part of the protected program execution, and then read back again at another point, as another means of confirming the presence of the dongle.

Demo Mode

Many clients place their software in 'demo' mode if the security dongle is not found. Clients are then encouraged to make copies of the application to distribute to their friends. This provides a great marketing strategy that pays off with additional sales.

Product Version Control

This section addresses the issue of using the KEYLOK dongle to assure that your end-user purchases product upgrades.

Reserve one or more addresses within the KEYLOK dongle memory for writing license information regarding which products and/or features are accessible to the user. When you create a product upgrade, change the required value in the reserved memory location that is necessary to work with the upgrade. For example, the prior product release might be version 6 and the number 6 is programmed into the dongle memory for version control. The software upgrade reads this memory location, and if it is not 7 or larger (the current version) the program refuses to execute and provides the user with appropriate instructions for acquiring permission to use the upgrade.

Two techniques can be used to update the dongle memory.

- Using the **Remote Update** capabilities of the KEYLOK system the memory containing the revision control code can be updated via an email exchange or a phone conversation with your client. Refer to the ***Remote Update*** chapter of this manual for additional information.
- Another technique would be to send out a new security dongle with the software upgrade. The dongle would be programmed with the appropriate authorization for use with the upgrade.

Anti-Debugging Utility

PPMON.EXE is a utility that prevents a debugger from being attached to your executing program. The Anti-Debugger is activated by calling the KEYBD(OFF) function. Although the function name implies that the keyboard is turned off, the anti-debugging utility *PPMON* is launched. This adds much greater security to your protected program. PPMON64.EXE is used for 64-bit application running on 64-bit architecture. The calling of PPMON.EXE/PPMON64.EXE is handled automatically.

KEYLOK API Reference

In this chapter we describe the KEYLOK Application Programming Interface (API).

~~All KEYLOK hardware can be accessed through one common interface. Whether you access the USB, parallel or serial port dongle the function calls are identical. Therefore you can access any of the KEYLOK products using the same source code.~~ All KEYLOK hardware can be accessed through one common interface. Therefore, you can access any of the KEYLOK products using the same source code.

The KEYLOK API has these important advantages:

- Easy-to-use function calls
- Local access through ~~LPT, RS232 serial and~~ USB ports
- Remote access via TCP/IP
- Hardware access via device drivers (drivers NOT required for KEYLOK3 or KEYLOK Fortress) for Windows 9x/ME/NT/2000/XP/Vista/7/8/10/11/ Server2003 / Server2008/Server2008R2/Server2012
- ~~Hardware access via standard RS232 calls for any computer or OS with RS232 support for serial port dongles.~~

The KEYLOK API is Easy, Secure and Portable

Overview

Access to the KEYLOK API is accomplished by, either linking with one of the KEYLOK library (LIB) files or utilizing the KEYLOK DLL (KL2DLL32.DLL/KL2DLL64.DLL). If a DLL is required for your development language the appropriate DLL(s) will be included in the same directory as your sample source code. Some development languages and operating system versions support finding the DLL in the same directory as the application, but many require that you copy the DLL to the ~~\Windows or~~ \Windows\system32 or \Windows\SysWOW64 directory.

Most KEYLOK API calls are made with the exposed *KFUNC* function. For ease and readability this function has been encapsulated into a function *KTASK* within the sample code.

The following calling sequence is required for most development languages. Check the KTASK function in the demonstration program for the appropriate calling sequence for your language/compiler/environment.

Formatted: Not Highlight

Command Code - KFUNC(Command Code, Arg2, Arg3, Arg4)

Argument requirements vary by function (see function description for details). The first argument is the **Command Code** and is used to select the task to be performed, as follows:

CAUTION: Some languages require special care when generating arguments to assure that an illegal value is not assigned to the argument. An example would be attempting to assign the value 40,000 to a signed integer argument that can only have values between -32,768 and 32,767. The sample programs demonstrate how to handle these situations. An examination of function 'KTASK' in the demonstration program will provide an example of the proper calling sequence.

Check for KEYLOK

A successful ***Check for KEYLOK*** process is a prerequisite to running any other security dongle task (~~e.g.~~e.g., reading or writing memory, etc.). Many companies are content to use only the ***Check for KEYLOK*** process for protecting their software. This task involves verifying that a dongle built uniquely for your company is present.

CHECK FOR KEYLOK

All other security dongle functions MUST be preceded by a successful **Check for KEYLOK** event. The **Check for KEYLOK** involves an exchange of information between the host and the security system using a different series of bytes each time the dongle is interrogated (~~i.e.~~i.e., using an active algorithm). This task requires two sequential calls to *KFUNC,* as follows:

Check For KEYLOK (First Call)	
Prerequisite	None
Argument1	**KLCHECK = 1**
Argument2	**Validate Code 1**
Argument3	**Validate Code 2**
Argument4	**Validate Code 3**
Return Values	Each of the return arguments (ReturnValue1 and ReturnValue2) from this first call must be manipulated to create the arguments sent during the second call.

Check For KEYLOK (Second Call)	
Prerequisite	This call must be immediately preceded by the first call of the ***Check for KEYLOK*** sequence
Argument1	Exclusive OR (XOR) the constant **ReadCode3** with ReturnValue2 and then XOR the resulting value with the value created by rotating the bits in ReturnValue1 left by a rotation count value established by ANDing ReturnValue2 with 7.
Argument2	The value created by rotating the bits in ReturnValue2 by a rotation count value established by ANDing ReturnValue1 with 15.
Argument3	The value created by XORing ReturnValue1 and ReturnValue2.
Argument4	Dummy argument. Whenever a dummy argument is called for, you can substitute any value (e.g., either zero or a random number).
ReturnValue1	ReturnValue1 = **ClientIDCode1**

ReturnValue2	ReturnValue2 = **ClientIDCode2**

If both parts of the customer identification code (**ClientIDCode1** and **ClientIDCode2**) are successfully retrieved from the dongle, then you have the option of proceeding with other dongle operations.

NOTE: Each time a ***Check for KEYLOK*** is performed the *ReadAuthorization* and *WriteAuthorization* flags are reset (~~i.e.~~i.e., deactivated). This means that the authorization sequence must be re-sent prior to any memory read/write operation.

Get Dongle Type

This task retrieves the type of dongle being communicated with.

Get Dongle Type (GETDONGLETYPE = 33)	
Prerequisite	Successful ***Check for KEYLOK***
Argument1	**GETDONGLETYPE = 33**
Argument2	Dummy argument*
Argument3	Dummy argument*
Argument4	Dummy argument*
ReturnValue1	Valid Values: 1 = FORTRESS 2 = KEYLOK3 3 = KEYLOK2
ReturnValue2	Undefined

***It is recommended that a random number be passed for dummy arguments.**

Read Operations

Upon successful completion of the ***Check for KEYLOK*** you have the option of adding additional security to your program by making use of either the dongle serial number (unique to each dongle) and/or the programmable memory within the dongle. Examples of what can be done with the dongle memory include:

- Writing a random value to the dongle and reading it back later to confirm dongle presence
- Storing a copyright message or the actual name of your client within the dongle memory and reading or displaying this information from within your program,
- Storing critical constants used in program calculations within the dongle,
- Storing licensing information used to enable which of multiple products sold by your company can be executed, or alternately which features within an application can be executed,
- Storing a count of the number of uses available (counters)

Read Authorization

Prior to running any read-related task, a successful ***Check for KEYLOK*** must be completed. The first read-related task that MUST be executed is ***Read Authorization***; successful completion of this task authorizes the dongle to perform memory read operations. If the dongle does not receive the correct Read password subsequent read operations retrieve random information. It is not necessary to perform the ***Read Authorization*** call unless you intend to read the dongle serial number or other memory within the dongle after completing the ***Check for KEYLOK***.

Read Authorization (READAUTH = 2)	
Prerequisite	Successful ***Check for KEYLOK***
Argument1	**READAUTH = 2**
Argument2	**ReadCode1**
Argument3	**ReadCode2**
Argument4	**ReadCode3**
ReturnValue1	None
ReturnValue2	None

NOTE: After a ***Read Authorization*** there is no indication of success or failure of the operation. We intentionally avoid a success/failure code ~~in order to~~to complicate the task of someone writing a program to simply test all possible combinations until a success flag is returned. The only way to know that you have had success is to read some memory value (~~e.g.~~e.g., serial number, etc.) twice and confirm the same number was received both times. However, from a practical perspective, if you have run a successful ***Check for KEYLOK*** with the dongle and you have sent it the proper authorization codes, then the return values from read operations will be correct. If you fail in any of the ~~prerequisites~~ prerequisites, then the return values from read operations will be random numbers. The same discussion is applicable to the write authorization command discussed later in this manual.

Read Serial Number

This task retrieves the unique serial number programmed into the security dongle. No two dongles with the same company-unique information will contain the same serial numbers.

Read Serial Number (GETSN = 3)	
Prerequisite	Successful ***Read Authorization***
Argument1	**GETSN = 3**
Argument2	Dummy argument*
Argument3	Dummy argument*
Argument4	Dummy argument*
ReturnValue1	The dongle serial number
ReturnValue2	Undefined

*It is recommended that a random number be passed for dummy arguments.

Read Customer Serial Number

This task retrieves the custom serial number programmed into the security dongle. The value is set using the SETLONGSN API call.

Read Custom Serial Number (GETLONGSN = 31)	
Prerequisite	Successful ***Read Authorization***
Argument1	**GETLONGSN = 31**
Argument2	Dummy argument*
Argument3	Dummy argument*
Argument4	Dummy argument*
ReturnValue1	The custom dongle serial number
ReturnValue2	Undefined

*It is recommended that a random number be passed for dummy arguments.

Read Memory

This task allows you to retrieve information written into the programmable memory. Remember that the 112 bytes of memory are partitioned into 56 addressable memory cells (addresses 0-55).

Read Memory (GETVARWORD = 4)	
Prerequisite	Successful ***Read Authorization***
Argument1	**GETVARWORD = 4**
Argument2	Desired address (0-55)
Argument3	Dummy argument*
Argument4	Dummy argument*
ReturnValue1	The memory contents
ReturnValue2	Undefined

*It is recommended that a random number be passed for dummy arguments.

Write Operations

Memory within the dongle can be used to store information related to your program. This can either be 'static' or 'dynamic' information. An example of static information would be something you write into the dongle before you ship it to your client, which is subsequently read while attached to your client's computer. An example of dynamic information would be something that is written into and read from the dongle while it is attached to your client's computer. An example would be a counter.

In ~~genera~~general, the prerequisites to memory write operations are:

1) Successful ***Check for KEYLOK***
2) Successful ***Read Authorization***
3) Successful ***Write Authorization***

Write Authorization

Successful completion authorizes dongle to perform memory write operations. If the dongle does not receive the correct Write password the dongle will ignore write operations.

Write Authorization (WRITEAUTH = 5)	
Prerequisite	Successful ***Read Authorization***
Argument1	**WRITEAUTH = 5**
Argument2	**WriteCode1**
Argument3	**WriteCode2**
Argument4	**WriteCode3**
ReturnValue1	Undefined
ReturnValue2	Undefined

Write a Variable Word

This task is used to modify the contents of programmable read/write memory within the KEYLOK.

Write Variable Word (WRITEVARWORD = 6)	
Prerequisite	Successful ***Write Authorization***
Argument1	**WRITEVARWORD = 6**
Argument2	Target address (0-55)
Argument3	Desired contents
Argument4	Dummy argument*
ReturnValue1	Undefined
ReturnValue2	Undefined

*It is recommended that a random number be passed for dummy arguments.

Set Customer Serial Number

This task is used to modify the contents of 32-bit programmable customer serial number within the KEYLOK.

Set Customer Serial Number (SETLONGSN = 31)	
Prerequisite	Successful ***Write Authorization***
Argument1	**SETLONGSN = 31**
Argument2	Upper 16-bit value
Argument3	Lower 16-bit value
Argument4	Dummy argument*
ReturnValue1	Undefined
ReturnValue2	Undefined

*It is recommended that a random number be passed for dummy arguments.

Decrement A Counter

This task is used to decrement the contents of a memory location. This is useful when a particular memory word is being used as a counter. The calling program receives the result of the decrement process by return of an ERROR code.

Decrement Counter (DECMEMORY = 7)	
Prerequisite	Successful ***Write Authorization***
Argument1	**DECMEMORY = 7**
Argument2	Target address (0-55)
Argument3	Dummy argument*
Argument4	Dummy argument*
ReturnValue1	The number of counts remaining if no error was encountered
ReturnValue2	ERROR condition codes: 0 No error 1 Memory has already been counted down to zero no remaining counts 2 Invalid address requested 3 Write authorization not provided must 'Write Authorize' before attempting to decrement memory

*It is recommended that a random number be passed for dummy arguments.

Date Control Tasks

The following tasks are used to control an expiration date for use in your product. They use reserved dongle memory, within which the last valid system date and time and an expiration date are written. Each time a successful call is made to compare the current date to the expiration date the most recent date and time information is refreshed within the dongle. If the date or time has been set ~~back~~back, then you can disable your software from operating until the clock has been properly reset. You may wish to utilize a counter in conjunction with this ~~function, and~~function and take more drastic action if the clock has been found set back more than once.

If your client is leasing your software or you have established a demonstration time period, then the remote update tasks described in the next section provide an ideal means of extending the expiration date. (You can also use the ***Remote Update*** utilities supplied by KEYLOK.) Further, if you embed the remote tasks in an application that also checks for the expiration date, then it is possible to force the client to have their~~his~~ system clock set properly, because unless their~~his~~ system date is correct (~~i.e.~~i.e., matches yours) the remote tasks will not operate.

Formatted: Font color: Auto

If an end-user runs your expiration date protected program while the clock is set ahead, then the last-use date/time will be set into the future and they will no longer be able to run your software unless 1) your program expiration date is later than the date they set the computer to, and they set the computer date forward to the date/time that it was set to at the time they last ran your program, or 2) the last use date/time stored in the dongle is reset. The recommended solution to resolve this problem is to provide your end-user with remote update capability. When you perform a remote update using the extend expiration date task, the last use date/time will be reset to the current date/time on the end-user's computer. This is safe because remote updates will only work if the end-user's computer is set to the same date as your computer. Also, the extend expiration task accepts a value of zero months for the amount of extension. Therefore, the net ~~affect~~effect of performing this function is to simply reset the last use date/time, with no impact upon the expiration date setting.

When you set the expiration date within a security dongle, the last usage date/time is reset to the current system date/time on the computer on which the expiration date is programmed. This feature is useful for resetting last use date/time stored information if it becomes corrupted ~~as a result of~~because of someone accidentally/intentionally setting his or her system date/time ahead. This often happens to our clients when testing the expiration date features of KEYLOK.

CAUTION: When testing the lease expiration date related functions, it is important that you do not use an expiration date or system clock setting that is prior to the current year.

Get Lease/Demo Expiration Date

This task is used to read the expiration date. This date is used for comparison to the current system date as a means of establishing the remaining time period.

Get Expiration Date (GETEXPDATE = 8)	
Prerequisite	Successful ***Read Authorization***
Argument1	**GETEXPDATE = 8**
Argument2	Dummy argument*
Argument3	Dummy argument*
Argument4	Dummy argument*
ReturnValue1	The expiration date is encoded in the following bit format: YYYYYYYMMMMDDDDD, where YYYYYYY + Base Year (~~i.e.~~i.e., 1990) = Year MMMM = Month of Year (1 = January) DDDDD = Day of Month (1-31) The sample code shows how to encode the date.
ReturnValue2	Undefined

TIP: Refer to the sample code for a better understanding of how to format the arguments.

***It is recommended that a random number be passed for dummy arguments.**

Check Lease/Demo Expiration

This task is used to compare the expiration date stored in the KEYLOK memory with the current date as read from the system clock. The purpose of the comparison is to establish ~~whether or not~~whether the expiration date has been reached. This task also refreshes the last known valid date and time stored in the dongle ~~as long as~~if the current date and time are more recent. Any attempt on the part of the end-user to set back his clock will result in an error when running this task.

Check Lease Date (CKLEASEDATE = 9)	
Prerequisite	Successful ***Write Authorization***
Argument1	**CKLEASEDATE = 9**
Argument2	Dummy argument*
Argument3	Dummy argument*
Argument4	Dummy argument*
ReturnValue1	Computer's current System Date in the format YYYYYYYM MMMDDDDD (Where YYYYYYY + 1990 = Year) MMMM = Month of Year (1 = January) DDDDD = Day of Month (1-31)
ReturnValue2	Status Code - Result of comparison -2 = Lease expired (clock date greater than lease expiration date) -3 = System date has been set back -4 = No lease date (DateAddress contains '0'. This is the return code to be expected if a check is made with an as-manufactured dongle, to which you have not yet sent a desired expiration date) -5 = Invalid lease expiration date -6 = Last date system used is corrupt - unable to write. This error means that the dongle is not functioning properly. Either it has been hit by lightning, etc. and the memory has been altered, or some electrical failure occurred during memory write that prevented writing the correct value to dongle memory. This error code is used primarily for internal debugging purposes to identify 'bugs' in our code associated with the encryption/decryption/ update of this information. What is being stored in this memory is the last date on which a successful check-expiration-date task was performed. The value 'date' is only allowed to march forward. Only a trusted person can clear this error. Each time you execute the set-expiration-date task, the current system date (on the computer on which the dongle programming is being done) is written into this memory within the dongle. +n = Approximate number of days until lease expires.

TIP: Refer to the sample code for a better understanding of how to format the arguments.

*It is recommended that a random number be passed for dummy arguments.

Set Lease/Demo Expiration Date

This task is used to initialize an expiration date. This date is used for comparison to the current system date as a means of establishing the remaining time period.

Set Expiration Date (SETEXPDATE = 10)	
Prerequisite	Successful ***Write Authorization***
Argument1	**SETEXPDATE = 10**
Argument2	Dummy argument*
Argument3	The expiration date is encoded in the following bit format: YYYYYYYMMMMDDDDD (where YYYYYYY + Base Year (~~i.e.~~i.e., 1990) = Year MMMM = Month of Year (1 = January) DDDDD = Day of Month (1-31) The sample code shows how to encode the date. A value of zero for this argument will result in the expiration date check being disabled.
Argument4	Dummy argument*
ReturnValue1	Undefined
ReturnValue2	Undefined

TIP: Refer to the sample code for a better understanding of how to format the arguments.

***It is recommended that a random number be passed for dummy arguments.**

Network Control Tasks

The following tasks are used to set or get the number of simultaneous authorized users of your application installed on a network. We provide a server application that communicates with the KEYLOK dongle installed on the machine on which the server application is running. This technique works on any network operating system that has active support for the TCP/IP protocol. A protected program running on any node on the network can then access the dongle through the server application, up to the maximum simultaneous user count programmed into the dongle.

See section ***Dongle Access Over a Network*** of this manual for details related to using KEYLOK security with networks.

Set Max User Count

This task initializes the count of authorized simultaneous network sessions.

Set Max User Count (SETMAXUSERS = 11)	
Prerequisite	Successful ***Write Authorization***
Argument1	**SETMAXUSERS = 11**
Argument2	The desired simultaneous session count (less than or equal to the limit set in the dongle hardware)
Argument3	Dummy argument*
Argument4	Dummy argument*
ReturnValue1	Undefined
ReturnValue2	Undefined

*It is recommended that a random number be passed for dummy arguments.

Get Max User Count

This task retrieves the maximum number of authorized simultaneous network session counts.

Get Max User Count (GETMAXUSERS = 12)	
Prerequisite	Successful ***Read Authorization***
Argument1	**GETMAXUSERS = 12**
Argument2	Dummy argument*
Argument3	Dummy argument*
Argument4	Dummy argument*
ReturnValue1	The maximum simultaneous network session count programmed into the dongle.
ReturnValue2	Undefined

*It is recommended that a random number be passed for dummy arguments.

Get Current Number of Users

This task retrieves the current number of active sessions and the maximum authorized simultaneous network user count. A similar functionality is available in the form of a utility, *NetKeyMonitor.exe*. See section 5.1 of this manual for further details regarding this utility.

Get Network Counts (GETNWCOUNTS = 20)	
Prerequisite	Successful ***Read Authorization***
Argument1	**GETNWCOUNTS = 20**
Argument2	Dummy argument*
Argument3	Dummy argument*
Argument4	Dummy argument*
ReturnValue1	The current number of active sessions communicating with the security dongle.
ReturnValue2	The maximum number of authorized simultaneous network sessions programmed into the dongle.

*It is recommended that a random number be passed for dummy arguments.

Do Remote Update

This task retrieves the AUTHORIZE.DAT file and processes the remote update against the dongle. The serial number of the dongle must match one of those within AUTHORIZE.DAT A similar functionality is available in the form of a utility, *RemoteUpdateEmailUser.exe*. See the Remote Update section of this manual for further details regarding this utility.

Do Remote Update (DOREMOTEUPDATE= 21)	
Prerequisite	Successful ***Write Authorization***
Argument1	**DOREMOTEUPDATE = 21**
Argument2	Dummy argument*
Argument3	Dummy argument*
Argument4	Dummy argument*
ReturnValue1	Undefined
ReturnValue2	Undefined

*You can pass dummy arguments and RemoteUpdateEmailUser.exe will look for AUTHORIZE.DAT in the directory of the executable. However, you can set Argument2 = 1357 and then you can pass a pointer to the ANSI string containing the path and filename you want to process in Argument4. If you are building a 64-bit app, you will need to use Argument3 to pass the high address and use Argument4 for the low address. It is recommended that a random number be passed for dummy arguments if you are not setting the name and path of the AUTHORIZE.DAT file.

Get Absolute Max User Count

This task retrieves the absolute maximum number of simultaneous users set in the hardware of a multi-user network dongle. This can be used to identify a networking dongle from a non-networking dongle.

Get Absolute Max User Count (GETABSOLUTEMAXUSERS = 32)	
Prerequisite	Successful ***Write Authorization***
Argument1	**GETABSOLUTEMAXUSERS = 32**
Argument2	Dummy argument*
Argument3	Dummy argument*
Argument4	Dummy argument*
ReturnValue1	The number of allowed TCP/IP users built into the dongle.
ReturnValue2	Undefined

Examples:

KTASK(GETABSOLUTEMAXUSERS, dummy, dummy, dummy)

A 25-user TCP/IP dongle would return 25.

Demo dongles and single-user dongles will return 1.

*It is recommended that a random number be passed for dummy arguments.

Dongle Communications Termination

This task is used to close a session between the application and the dongle.

Terminate

Terminate (TERMINATE = -1)	
Prerequisite	None
Argument1	**TERMINATE = -1**
Argument2	Dummy argument*
Argument3	Dummy argument*
Argument4	Dummy argument*
ReturnValue1	Undefined
ReturnValue2	Undefined

*It is recommended that a random number be passed for dummy arguments.

Extended Fortress Functions

These two additional Fortress functions are separate from the KFUNC API calls. They are separate functions and require separate declarations.

~~*A Block*~~*A Block* *Read / Block Write*

These functions are designed to allow quick and efficient reading/writing of large amounts of data to/from the dongle without incurring significant overhead for each word of data to be exchanged The first argument to the KBLOCK function is the Task It is used to identify whether you are reading or writing a block of memory The second argument 'DongleMemoryAddress' is the starting physical address of the word within the dongle at which reading/writing is to begin The third argument is the number of words of memory to be read/written The fourth argument is a pointer to the array containing the data to be written, or the pointer to the array to receive the data read from the dongle.

KBLOCK(Task, Address, WordLength, pData)	
Prerequisite	Read Authorization for BLOCKREAD and Write Authorization for BLOCKWRITE
Task	**BLOCKREAD = 84 / BLOCKWRITE = 85**
Address	Starting word address of the memory within the dongle.
WordLength	The number of words of memory to be read/written **NOTE:** One word can contain two characters of text, if desired.
pData	Pointer to the array that contains data to be written to the dongle, or to receive data read from the dongle.

~~*B Execute*~~*B Execute* *Code* ~~*On*~~*on* *Dongle KEXEC*

This task provides you with the ability to execute your code directly on the Fortress dongle, which offers significant security advantages The code is only executed on the dongle and data is passed between your application and the dongle using a 250 byte buffer The code on the dongle cannot be inspected by a would-be hacker, thus providing the ultimate in security for special algorithms that make your protected application particularly valuable, and without which your program will not perform its expected functionality. Your function(s) is loaded onto the dongle as an individual program(s) in a directory structure

The first argument is a pointer to a string containing the name of the folder in the directory structure that contains your code and optionally a data file(s) that will be used by your code. The name of this folder is assigned by KEYLOK personnel for your company

The second argument is a pointer to a string containing the name of the program that contains your company unique code. One or more programs can be stored within the dongle to execute your application unique algorithms. The program file name(s) are assigned by KEYLOK personnel.

The third argument is a pointer to a string that contains an eight-character (64 bit) password that must be known to execute programs stored in your company folder on the dongle. This value can be changed from the default value assigned by KEYLOK personnel to whatever value you wish

The fourth argument is a pointer to a buffer that is used for passing arguments to/from your company unique functions.

The fifth argument is the size of the buffer in bytes (unsigned short) containing IN/OUT arguments (maximum of 250 bytes).

KEXEC (LPSTR ExeDir, LPSTR ExeFile, LPSTR UserPIN, LPSTR Buffer, USHORT BufferSize)	
Prerequisite	Successful ***Check for KEYLOK***
ExeDir	Directory name on the dongle
ExeFile	File name of the executable on the dongle
UserPIN	Security PIN which allows access to execute code on dongle
Buffer	Data buffer for passing data between the application and the dongle
BufferSize	Size of data buffer in bytes

Please see the ***KEYLOK*** Code ~~On~~on Dongle Manual for details on the type of code you can transfer to the dongle.

Commented [RB4]: I don't remember seeing this manual

~~C RTC~~C RTC Functions

Fortress Rechargeable Real Time Clock is an excellent additional protection to secure software independent of a system clock. This would prevent users from attempting to manipulate system time to circumvent licensing. Simple API calls can be made to check the RTC and return it to your software. The second return value contains the time in milliseconds which must be added to the base date January 1st ~~1990~~, 1990.

CheckRTC()	
Prerequisite	Successful ***Read Authorization***
Argument1	**CKREALCLOCK = 82**
Argument2	Dummy argument
Argument3	Dummy argument
Argument4	Dummy argument
ReturnValue1	RTC time in milliseconds
ReturnValue2	RTC time in milliseconds

ReturnValue2 must be multiplied by 65536 and then summed with ReturnValue1 to obtain the total number of milliseconds.

Anti-Debugging Utility

The **KEYBD(OFF)** function is used to activate the anti-debugging utility **PPMON.EXE**. *PPMON.EXE* is a utility that prevents a debugger from being attached to your executing program. Although the function name implies that the keyboard is turned off, actually the anti-debugging utility PPMON is launched. This adds much greater security to your protected program. The anti-debugger runs automatically with some functions such as RemoteUpdate.

KEYBD(0) - This call launches the anti-debugging utility.

This call need only be performed one time from within the protected application. In general, this call can be placed anywhere in your application, however there are instances in which this call must be placed outside of the InitApplication() function.

Dongle Access Over a Network

Overview

General Information

The KEYLOK security system can be used to share a single dongle among various applications running on a network. The advantage of this technique is that multiple copies of a protected application running on different computers can be controlled through use of a single security dongle.

A server application is provided that communicates with the KEYLOK dongle installed on the machine on which the server application is running (see Fig. 2). This technique should work on any network operating system running the TCP/IP protocol, which is currently the most widely supported network protocol. A protected program running on any node on the network can then access the dongle via the network, up to the established maximum simultaneous user count. (**NOTE:** See later section "Controlling Authorized User Count" for more details.)

Networking Components

The following components are required to implement the network security system. Each of these components is installed and configured automatically as appropriate by the KEYLOK install utility.

Server Application: This is the program that each copy of your protected application communicates with in order to acquire remote contact with the security dongle. This program acts as the interface between your protected application and the security dongle. The name of the server application is *KLSERVER.EXE*. The server runs as a Windows service. Up to three copies of this server can be run on a network. Note that the total number of sessions that can be run simultaneously on a network equals the sum of the allowed sessions programmed into each of the server dongles running on the network.

NOTE: The server application (KLSERVER.EXE) can be installed and run on any machine running Windows; it does not require a server OS.

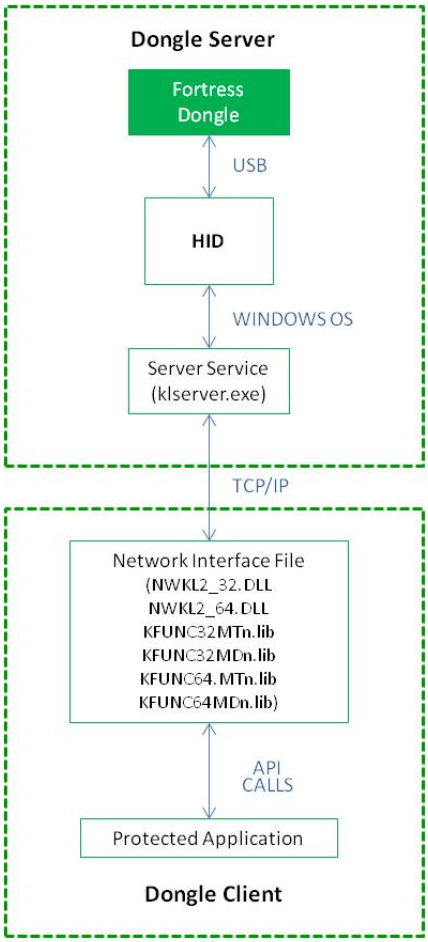

Figure 2: Dongle Access over a TCP/IP network.

You can optionally set a fixed timeout (in minutes) for dongle client sessions connected to a server by including a file named *KLTCPIP DAT* in the same directory as *KLSERVER EXE.* This is intended to recover sessions that have inadvertently been lost in such a way that the application and/or OS did not send a signal to the server application to close the session. Once a session 'times-out' the session is then freed for access from any other client. The file should be a standard ASCII text file containing the number of minutes that can pass with no activity from the client before a session times out. The default timeout period for a session with no dongle activity is 9 hours (540 minutes). This is intended to correspond to an individual's full work shift of 8 hours including an hour off for lunch.

You can optionally set the port for dongle client communication by including a file named *KLSVRPORT.DAT* in the same directory as *KLSERVER EXE* and your client application. This is intended to specify a port different from the default of 4242.

You can optionally set IP address used by *KLSERVER.EXE* by including a file named *KLSVRIP.DAT* in the same directory as *KLSERVER EXE.* This is intended to be used when the server has multiple network cards.

KEYLOK security dongle: The security dongle required for network operation is identical to that used for standalone operation except that for more than one simultaneous user you must use special multi-session versions of our dongles. The dongle must be physically installed on the network node that is running the server application *KLSERVER EXE.* The dongle may be optionally programmed with user limits. If more than one key designed to respond to the same set of company unique codes is installed on the same ~~network~~network, they must be physically installed on two separate platforms.

Client Application: This is the protected program that must be capable of communicating with the security dongle over the network in order to confirm the presence of a proper dongle, and to be able to read and/or write to the security dongle memory. The protected application can also be run on the same computer platform as the server application.

Clients can be forced to use a specific server by a *TCPIPSVR DAT* file in the following location:

Architecture	32-bit OS Location	64-bit OS Location
32-bit	\Windows\System32	\Windows\SysWOW64
64-bit		\Windows\System32

The *TCPIPSVR DAT* file is an ordinary text file that contains the network name or IP address of the server to which you want the client to connect. Note that a blank *TCPIPSVR DAT* file is created by default during a ~~Client~~client ~~installation;~~installation: if a server name or IP address is specified during the installation, it will be written into the *TCPIPSVR DAT* file.

Network: The computers on which the protected application and server application is running must be physically connected via a network with the TCP/IP protocol supported on each platform. We strongly recommend that the server and all clients be in the same subnet; in some cases, it may be possible to access dongles across subnets.

Network Utilities

We provide three utilities to assist in implementing and monitoring networked dongles. Note that all of these utilities are company specific (i.e. i.e., the 'DEMO' dongle version will not work with company unique dongles, and vice ~~versa) These~~versa) These utilities are as follows:

NetDemo is a network-enabled version of our demonstration program. NetDemo illustrates nearly all of the capabilities of KEYLOK dongles, and may be used to check dongle serial numbers, check or set the contents of dongle memory (one 2-byte location at a time), check or set lease expiration dates, or, on multi-user network dongles, to check or set the maximum number of simultaneous users (up to the limit of the dongle hardware). **NOTE:** This utility should ***not*** be sent to end-users.

NetKeyMonitor is a utility that can be run on the dongle server or client platform. The utility shows all active dongle servers detected on the network and reports the maximum number of allowed sessions as programmed into each corresponding security dongle, as well as the current number of active sessions on each dongle server (for your company-unique dongles)~~.~~ Its only function is to check for the presence of network dongles, so you can safely send it to your end-users as a diagnostic tool.

VerifyNetworkKey is a utility that allows you to make sure that a client on a network can communicate with a dongle mounted on a remote server. Its only function is to check for the presence of a network dongle, so you can safely send it to your end-users as a diagnostic tool.

Network Protected Program Preparation

The primary difference between a protected program designed to look for the dongle locally and the network version is the interface file with which the application is linked, as follows:

- If a 32-bit application is linked with *KFUNC32xx lib*, checks for the dongle will be made only on the local machine. If the application is linked with the network version of the object file, *KFUNC32xxN lib*, then checks will be made both locally and on the network. Successful linking of 32-bit applications with *KFUNC32xxN lib* also requires the WIN32 SDK supplied libraries of *NETAPI32 LIB and WSOCK32 LIB*.

- Similarly, if the application normally uses a DLL for dongle communications, then the non-network DLL must be replaced with the network version of the DLL. For 32-bit applications, replace *KL2DLL32 DLL* (for local dongle checks) with *NWKL2_32 DLL* (for network checks).

The network interface files (NWKL2_32 DLL or KFUNC32xxN LIB) first search the local machine for a dongle. Then, if a local dongle is not found, they search the network. The specific search order is:

1) Search for local USB dongle
~~2) Search for local parallel dongle~~
2) Search for TCP/IP network key for remote dongle over the network

Note that the non-network search is as follows:

1) Search for local USB dongle
~~2) Search for local parallel dongle~~

Controlling Authorized User Count

There are two methods that can be used to control the maximum number of simultaneous users of your application on the network, as follows:

- Dongle Limited (Absolute Maximum User Count): Multi-user KEYLOK dongles are pre-programmed with an absolute maximum number of simultaneous users that can be supported. This limit is physically set in the dongle and cannot be changed. Single-user dongles are accessible over a ~~network, but~~network but will only support one user at a time.

- Programming Limited (Maximum Authorized Users): The KEYLOK dongle can be programmed with the maximum number of sessions you wish to authorize with the dongle (API SETMAXUSERS). This number can be set to any value from 1 up to the dongle-limited maximum number pre-programmed into the dongle. **NOTE:** Fortress dongles have a maximum allowed session limit of 127.

The maximum number of sessions of a network-enabled application that can be run simultaneously is equal to the total number of sessions programmed into all dongles accessible on the network.

When using a 32-bit application linked with the **network version** of the security system each active session that communicates with the dongle **is counted** against the user limit. When an application linked with the **standalone version** of the security system is used, then sessions of that application run on any machine including the one running the server application **will not be counted** against the session limit. There is no limit on the number of simultaneous sessions that can be run on a single machine when linking with the standalone version.

Installing the Server Application

The server application *KLSERVER EXE* must be installed and started on the platform on which the KEYLOK security dongle is attached. This is accomplished by running the ***INSTALL*** program with the Server checkbox checked or from the command line with /NF option. Please review the detailed instructions for running *INSTALL EXE* in the section of this manual entitled ***Distributing Your Application.*** Keep in mind that the dongle-server need not be running a Server OS.

The Client Application

Your application running on a target platform will have the necessary code to communicate with a remote dongle fully integrated into the application, provided that the application is linked with the network enabled version of the libraries that we provide.

Remote Update

KEYLOK provides two sets of **Remote Update** utilities that will allow you to securely read, write, or modify any of the programmable features of KEYLOK dongles remotely.

E-Mail Remote Update

The **E-Mail Remote Update** utility consists of two modules:

1) the software developer module (RemoteUpdateEmailDev exe)
2) the end-user module (RemoteUpdateEmailUser exe)

When a remote update is required, the developer runs the developer module (RemoteUpdateEmailDev exe) to generate an update file, AUTHORIZE DAT, and sends the end-user a copy of the end-user E-Mail Remote Update module (RemoteUpdateEmailUser exe) and a copy of the AUTHORIZE DAT file. The end-user runs the module and processes the received file against the dongle to update it.

There are various options for performing this task:

1) the end user can generate a REQUEST DAT file by running the RemoteUpdateEmailUserRequest exe utility and send this file to developer as an 'aid' in creating the AUTHORIZE DAT file. In this case each requested action is displayed to the developer for authorization, and the developer is also allowed to add additional update tasks.
2) the developer utility (RemoteUpdateEmailDev exe) can be run in GUI mode as described below,
3) the developer utility can be data driven by a file named 'DeveloperRemoteUpdate dat', in which case all of the desired update actions are written into a text file along with the target serial number(s) of the dongle(s) to be updated. The developer utility simply converts the input dat file into an encrypted AUTHORIZE DAT file. The utility can be run in quiet mode (i.e. i.e., no displayed messages [/q command line parameter activates quiet mode]) to further automate this task with a minimal of effort on the part of the developer.

A typical e-mail remote update session is as follows:

1) The developer runs the developer module *RemoteUpdateEmailDev exe.* When the program opens, the developer clicks ***Check for KEYLOK***, clicks OK, clicks Memory Read>Read Authorization, Memory Write>Write Authorization, and Email Remote Update>Software Developer. The developer is prompted to input the target dongle serial number(s). Up to 60 dongles can be updated from a single AUTHORIZE DAT file.

2) The developer module will then display a list of possible remote update tasks. The developer clicks the radio button next to the selected task, supplies any additional information needed (e g, memory address and value, new expiration date, etc.), and repeats this until all desired update operations have been selected. The developer than clicks "No More Requests," and the program responds with a message giving the serial numbers and update numbers for this update session. The update information is written to an encrypted *AUTHORIZE DAT* file.

3) The developer sends the end-user a copy of the end-user **E-Mail Remote Update** module *RemoteUpdateEmailUser exe* and a copy of the *AUTHORIZE DAT* file.

4) The end-user puts the *AUTHORIZE DAT* file and the end-user module into the same directory and runs the end-user module. When the program displays its window, the end-user clicks on the "Program Security Key" button. The program responds with a message saying that the security key has been updated ~~and also~~and displays the serial number of the dongle that has been updated.

Rather than using RemoteUpdateEmailDev exe, you can build your own end-user utility and use the API function DOREMOTEUPDATE (Value = 21) which will update the dongle with the AUTHORIZE DAT file located in the same directory as your utility. You will have to check for the dongle and get read and write authorization before processing this function.

CUSTDATA DAT and AUTHORIZE DAT

The *CUSTDATA DAT* file is used to keep track of how many update sessions have been conducted for a given dongle. For this reason, **it is critical that the CUSTDATA DAT file be backed up frequently and regularly.** A new *CUSTDATA DAT* file is created the first time that the developer module is ~~run, and~~run and updated by each run thereafter.

A new *AUTHORIZE DAT* file is generated for each update session and contains the actual encrypted update instructions used by the end-user module to update the dongle.

From a high level, the way that *CUSTDATA DAT* and *AUTHORIZE DAT* are used is as follows:

Each update to a customer will increase the "Update Number" within the *CUSTDATA DAT* file. The contents of the file are dongle serial numbers and update numbers, for example:

Serial #	Update #
1234	7
1235	0
1236	3
1237	1

When an *AUTHORIZE DAT* file is used to update a dongle, it first checks to see if the dongle serial number matches, and then compares the *AUTHORIZE DAT* update number with the number of the last update stored on the dongle (in encrypted form in a location not accessible to the user). This prevents the same update or older updates from being run on the same dongle twice. For example:

Before the first update:

	Serial Number	Update Number
AUTHORIZE DAT	1234	1
Dongle	1234	0

This will update the dongle and store the last update number 1 on the dongle.

After the first update, using the same *AUTHORIZE DAT*:

	Serial Number	Update Number
AUTHORIZE DAT	1234	1
Dongle	1234	1

This will not update the dongle because the last update number stored on the dongle is the same as the number in *AUTHORIZE DAT*.

To address situations in which the *CUSTDATA DAT* file is lost or corrupted, it is possible to re-generate/re-synchronize the files. For example, if the *CUSTDATA DAT* file in the above example was lost, it would be necessary to create an *AUTHORIZE DAT* with an update number of 2 or more. This can be done by resetting the version counter (click Email Remote Update>Erase Last File Version) and repeatedly generating new *AUTHORIZE DAT* files until the required sequence number is reached. Alternatively, you can use the utility named CustDataFileMaintenance exe to create and update CUSTDATA DAT.
NOTE: File regeneration only applies if the *CUSTDATA DAT* is lost. This file should be backed up often regeneration sets all update numbers to zero. It is then necessary to edit the file using the CustDataFileMaintenance exe utility for each dongle that has been sent remote update AUTHORIZE dat files.

Developer Remote Update DAT

The creation of AUTHORIZE DAT can be automated by providing RemoteUpdateEmailDev exe with a text file containing the tasks to be processed on the remote dongles.

DeveloperRemoteUpdate dat must be in the following format:

```
Count: 2
SerialNumbers: 1234, 5667
Task: 0
Address: 1
Value: 5
Task: 1
Address: 15
Value: 3
Task: 2
Address: 9
Value: 1
Task: 9
Address: 4000
Value: 12/28/2022
```

Count = the number of serials numbers to be processed. The maximum number in one DeveloperRemoteUpdate dat is 60
SerialNumbers = the serial numbers to be updated, separated by a comma and a space
Task = Task ID (see table below)
Address = memory address to be updated (see table below)
Value = value to be used for the task

Task	Task ID	Address	Value
Add to memory contents	0	Actual address	Positive integer
Extend expiration date “n” months	1	Any value (ignored)	Number of months
Bitwise “OR” of memory contents	2	Actual address	Any value
Replace memory contents	3	Actual address	Any value
Set maximum number of sessions	5	Any value (ignored)	1-127 see page 41
Set expiration date	9	Any value (ignored)	MM/DD/YYYY or 0 to remove expiration date

The maximum number of tasks which can be included in DeveloperRemoteUpdate dat is 60.

Distributing Your Application

Using the KEYLOK Install Utility

The easiest and recommended method to install the required KEYLOK interface files onto your end-users' computers is to use the KEYLOK Install Utility, *Install.exe*. This utility may be run from its own Graphical User Interface (GUI) or invoked, using command-line switches to select installation parameters, from your existing installation program such as InstallShield or WISE.

The installer determines which operating system is running, and then copies the appropriate individual driver and/or KEYLOK files to the proper directories.

NOTE: Modification of the NT/2000/XP/Vista/7/8/9/10/11 system registry requires that a user with Administrator permissions runs the utility.

Install Utility Command Line Arguments

Install.exe Utility	
Delimiters	Valid command line delimiters are **'/'** or '****' or **'-'** A space is required before each delimiter and options cannot be combined (i.e., '/QN' is illegal, but '/Q /N' is legal).
/F	Install KEYLOK3 files only
/NF	Install KEYLOK3 TCP/IP server networking files to allow remote access to a KEYLOK3 dongle via a TCP/IP network.
/Q	Quiet mode install – displays only fatal errors
/S:<IP address>	Valid only for a network client install. Forces the client to attempt to connect to the dongle server at <IP address>. You may also specify the network name of the server instead of its IP address.
/U	Uninstall Removes all previously installed files

Manual Installation

In some cases, it may necessary or desirable to install the files manually rather than using our installer. The following tables indicate what needs to be done to install the interface files manually on a Windows system. For other configurations, please contact Technical Support.

All Installations

The following files must always be installed, regardless of dongle type or network use.

Architecture	File Name	Description	32-bit OS Location	64-bit OS Location
32-bit	KL2DLL32.DLL	32-bit KEYLOK DLL	\Windows\System32	\Windows\SysWOW64
32-bit	PPMON.EXE*	Anti-debugger	\Windows\System32	\Windows\SysWOW64
64-bit	KL2DLL64.DLL	64-bit KEYLOK DLL		\Windows\System32
64-bit	PPMON64.EXE*	Anti-debugger		\Windows\System32

Network Server

Architecture	File Name	Description	32-bit OS Location	64-bit OS Location
All	KLSERVER.EXE	Network server service	\Windows\System32	\Windows\SysWOW64
32-bit	NWKL2_32.DLL	32-bit KEYLOK DLL	\Windows\System32	\Windows\SysWOW64
64-bit	NWKL2_64.DLL	64-bit KEYLOK DLL		\Windows\SysWOW64

The registry entry for the network server service is created by a Win32 call to *CreateService()*. This sets the *klserver.exe* service to start up automatically when the machine is started. The DLL files are only required on the network server if the client application is going to also be running on the network server machine.

Network Client

The following file must be installed where indicated in order for the protected application (if using DLL instead of linkable object) to be able to communicate with a dongle server.

Architecture	File Name	Description	32-bit OS Location	64-bit OS Location
32-bit	NWKL2_32.DLL	32-bit KEYLOK DLL	\Windows\System32	\Windows\SysWOW64
32-bit	PPMON.EXE*	Anti-debugger	\Windows\System32	\Windows\SysWOW64
64-bit	KL2DLL64.DLL	64-bit KEYLOK DLL		\Windows\System32
64-bit	PPMON64.EXE*	Anti-debugger		\Windows\System32

* Only required if the developer has chosen to implement calls to the anti-debugging utility **(strongly recommended).**

KEYLOK Utility Programs

KEYLOK supplies a number of utilities that you can use to program dongles, diagnose common driver, or network problems, and explore the capabilities of KEYLOK dongles. All of these utilities except *INSTALL.EXE* are company specific (i.e., the 'DEMO' dongle version will not work with company unique dongles, and vice versa). The utilities are:

WinDemo

WinDemo.exe is a pre-compiled version of our standard sample demonstration program. The source code for *WinDemo.exe* may be found in the ..\Samples\Windows\VisualStudio32 directory in our sample code. **WinDemo** illustrates nearly all of the capabilities of KEYLOK dongles, and may be used to check dongle serial numbers, check or set the contents of dongle memory (one 2-byte location at a time), check or set lease expiration dates, or, on multi-user network dongles, to check or set the maximum number of simultaneous users (up to the limit of the dongle hardware).

VerifyKey

VerifyKey.exe is a utility that allows you to make sure that the device drivers and interface files have been installed correctly so that your application will be able to communicate with a dongle. Its only function is to check for the presence of the dongle, so you can safely send it to your end-users as a diagnostic tool to make sure that the device drivers are working correctly.

KLTool

KLTool.exe is a very versatile ~~general purpose~~general-purpose programming and demonstration tool. In addition to all of the capabilities of **WinDemo**, it also allows reading and writing text strings to and from dongle memory, resetting of lease expiration dates (to no date set, so that the dongle never expires), programming of multiple keys using stored profiles.

NetDemo

NetDemo.exe is the network-enabled equivalent of **WinDemo**. It has the same functionality as **WinDemo**, except that it is designed to work with a dongle mounted on a remote dongle server that is accessed via a network.

VerifyNetworkKey

VerifyNetworkKey.exe, the network-enabled equivalent of **VerifyKey**, is a utility that allows you to make sure that a client on a network can communicate with a dongle mounted on a remote server. Its only function is to check for the presence of a network dongle, so you can safely send it to your end-users as a diagnostic tool.

NetKeyMonitor

NetKeyMonitor is a utility that can be run on the dongle server or client platform. The utility shows all active servers detected on the network and reports the maximum number of allowed sessions as programmed into the security dongle, as well as the current number of active sessions (for your company-unique dongles).

Install

INSTALL.EXE is the utility that installs the KEYLOK interface files and makes all necessary registry entries as appropriate for the type of installation selected. The utility will automatically detect the version of Windows on which it is running and install the appropriate versions of the interface files, anti-debugger and, if selected, networking files. **INSTALL.EXE** can be run either from its own GUI or from a command line. If run from a command line, it is possible to suppress all messages except fatal error messages.

CustDataFileMaintenance

CUSTDATAFILEMAINTENANCE.EXE is the utility that the developer uses to synchronize an authorize.dat file with custdata.dat. It is used to correct custdata.dat if it has gone out of sync with information stored in a target dongle. Lack of synchronization could be caused by 1) lost custdata.dat file, 2) experimentation with the remote update utilities by developers, 3) failure to apply previous remote update files by end-users, etc. Authorize.dat files that have an embedded remote update sequence number more than '3' greater than the last update to the dongle will not be processed, nor will they be processed if the authorize.dat file contains a sequence number less than or equal to that stored in the target dongle.

RemoteUpdateEmailDev

REMOTEUPDATEEMAILDEV.EXE is the utility that the developer uses to create authorize.dat files which are used to update the dongles remotely.

RemoteUpdateEmailUser

REMOTEUPDATEEMAILUSER.EXE is the utility that processes the authorize.dat on the client machine to update the dongle remotely.

RemoteUpdateEmailUserRequest

REMOTEUPDATEEMAILUSERREQUEST.EXE is the utility the end user runs to create a request.dat which is emailed to the developer and processed by RemoteUpdateEmailDev. Use of this utility is purely optional, and any actions converted to authorize.dat file contents require confirmation by the developer.

Using KEYLOK Under ~~Open Source~~Open-Source OS's[1]

We do not currently have a native LINUX install utility. To install on a Linux machine, go to the \SampleCode\Linux directory in the SDK ~~on the CD~~ and copy the USB directory to ~~some~~ a convenient directory on your Linux machine.

To implement KEYLOK protection in your Linux code, simply copy the appropriate sections of our Linux sample code into your application and link with our Linux object module, *KFUNC32 O.* We can also supply a shared object (.SO) version of our Linux object on request.

NOTE: that our Linux object also supports Java under Linux, using standard Java Native Interface (JNI) calls to *KFUNC* to access the dongle.

USB support is provided using the libusb ~~library, and~~library and assumes use of the usbdevfs file system.

The KEYLOK module *KFUNC32 O* is configured as a user-level driver, i.e., it accesses I/O ports directly rather than through a kernel driver. This has been done for simplicity, reliability, and speed. The user->kernel interface was deemed an all too easy hacking vulnerability. The downside of this is the application must run as root, or setuid root, in order to have permission to speak directly to I/O ports.

If you need to run as a non-root user, you can:

- Add a line to /etc/fstab right after the /proc entry:

  none /proc/bus/usb usbfs devmode=0666 0 0

  then, as root, issue the command

  umount /proc/bus/usb; mount -a

  This is wide open ~~access~~access, but permissions can be tuned with the busmode, listmode, devgid etc parameters. **NOTE:** you should specify 'usbfs' rather than 'usbdevfs'

- Use 'sudo' to grant the user permission to run the application with root privilege, without knowledge of the root password.

- Run the checker as a separate process or thread or fork the dongle checking process and communicate via shared memory (with the main process dropping root privileges), or check one first, then give up root privileges.

[1] E. G., OS A, LINUX, FreeBSD, QNX, etc.

QNX

If you are using the QNX operating system, our standard Linux sample code should run, with the following addition:

unsigned long userDelay = x;

where x is some ~~number This~~number. This will add a required delay to allow communication with the dongle.

Appendices

A. File Descriptions

Certain files are necessary in order for a protected application to access KEYLOK dongles. These files are automatically installed in the correct directories as appropriate for the type of installation by the KEYLOK installation utility. The files are described in the following table.

LIB files and DLLs	
KFUNC32MT.LIB KFUNC32MD.LIB	Library file to be linked into 32-bit applications to provide the communications interface to local (non-network) dongles. Linked with application
KFUNC32MTN.LIB KFUNC32MDN.LIB	Library file to be linked into 32-bit applications to provide the communications interface to a remote dongle via a network. Also provides support for locally mounted dongles. Successful linking of 32-bit applications with KFUNC32MTN.LIB or KFUNC32MDN.LIB also requires the WIN32 SDK supplied library of NETAPI32.LIB. Linked with application
KFUNC64MT.LIB KFUNC64MD.LIB	Library file to be linked into 64-bit applications to provide the communications interface to local (non-network) dongles. Linked with application
KFUNC64MTN.LIB KFUNC64MDN.LIB	Library file to be linked into 64-bit applications to provide the communications interface to a remote dongle via a network. Also provides support for locally mounted dongles. Successful linking of 64-bit applications with KFUNC64MDN.LIB or KFUNC64MTN.LIB also requires the WIN64 SDK supplied library of NETAPI64.LIB. Linked with application
KL2DLL32.DLL	32-bit DLL interface for locally mounted dongles. \Windows\System32 for 32-bit OS \Windows\SysWOW64 for 64-bit OS
NWKL2_32.DLL	32-bit DLL interface for local or remote (network) dongles. \Windows\System32 for 32-bit OS \Windows\SysWOW64 for 64-bit OS
KL2DLL64.DLL	64-bit DLL interface for locally mounted dongles. \Windows\System32 for 64-bit OS

NWKL2_64.DLL	64-bit DLL interface for local or remote (network) dongles. \Windows\System32 for 64-bit OS

Networking	
KLSERVER.EXE	KEYLOK TCP/IP server networking service that interfaces between your protected program and the device driver that actually communicates with the KEYLOK dongle. \Windows\System32 for 32-bit OS \Windows\SysWOW64 for 64-bit OS
KLTCPIP.DAT	An optional file used to set the maximum number of minutes a client session can be idle before being disconnected. \Windows\System32 for 32-bit OS \Windows\SysWOW64 for 64-bit OS
TCPIPSVR.DAT	An optional file used to force a network client to attempt to attach to a specific dongle server. Contains the IP address of the desired server. \Windows\System32 for 32-bit OS \Windows\SysWOW64 for 32-bit app on 64-bit OS \Windows\System32 for 64-bit app on 64-bit OS

Other Files	
PPMON.EXE	Anti-debugger routine for 16- or 32-bit systems. If a kernel-level debugger is detected, PPMON will terminate the application within 10 seconds of detection. \Windows\System32 for 32-bit OS \Windows\SysWOW64 for 64-bit OS
PPMON64.EXE	Anti-debugger routine for 64-bit systems. If a kernel-level debugger is detected, PPMON will terminate the application within 10 seconds of detection. \Windows\System32 for 64-bit OS

B. Protecting 64-bit Applications

In order to implement KEYLOK protection in a 64-bit application, it is only necessary to link your application with the 64-bit version of the interface files (library files or DLLs as appropriate for your programming language and development environment). No changes are necessary to your source code or to the KEYLOK API calls used.

The KEYLOK driver installer, **INSTALL.EXE**, will automatically detect a 64-bit operating system and install both the 64-bit and 32-bit interface files.

C. Special Language Support

There are two approaches used to support languages that do not have provisions for linkable object modules or the ability to utilize our DLL.

KEYLOK can supply a security routine in .EXE form. Shelling out of the protected application to run the executable security program and then returning back into the protected program provides security dongle communications. Communication between the protected program and the security module is accomplished through an intermediate data file, which can be read and written by both programs.

D. Technical Specifications

Standard USB

Environment	
Storage Temperature	-10° F to 175°F (-23° C to 80° C)
Operating Temperature	32° F to 157° F (+0° C to +70° C)
Dimensions / Connectors	
Fortress LS/RTC USB	50 x 18 x 8mm Standard Type A USB plug (1 1, 2 0 & 3 0)
Fortress RS USB	17 x 12 x 4mm Standard Type A USB plug (1 1, 2 0 & 3 0)
Memory	
Type	EEPROM
Data retention	At least 10 years
Programmable EEPROM	• 5,120 bytes = 2,560 memory locations • 18,944 bytes = 9,472 memory locations • 50,560 bytes = 25,280 memory locations • 1 Million Write cycles per location • Unlimited Read cycles
Security	
Encryption	Random Proprietary Encryption Algorithms
Authentication Password	2^{96} possibilities
Read Password	2^{128} possibilities
Write Password	2^{176} possibilities
Smartcard	ISO/IEC 15408 Security Evaluation Level EAL5+ Highest level globally
Power	
Power	No more than 15 milliamps / Battery on RTC lasts up to 3 years

Flash Drive USB

Environment	
Storage Temperature	-10° F to 175°F (-23° C to 80° C)
Operating Temperature	32° F to 157° F (+0° C to +70° C)
Dimensions / Connectors	
Fortress Flash Drive USB	50 x 18 x 8mm Standard Type A USB plug (1 1, 2 0 & 3 0)
Memory	
Type	Mass storage device (driverless) MLC, SLC / Erase Cycle MLC>10,000, SLC>100,000
Data retention	At least 10 years
Programmable EEPROM	• 5,120 bytes = 2,560 memory locations • 18,944 bytes = 9,472 memory locations • 50,560 bytes = 25,280 memory locations • 1 Million Write cycles per location • Unlimited Read cycles • Flash Capacity 1-16 GB
Security	
Encryption	Random Proprietary Encryption Algorithms
Authentication Password	2^{96} possibilities
Read Password	2^{128} possibilities
Write Password	2^{176} possibilities
Smartcard	ISO/IEC 15408 Security Evaluation Level EAL5+ Highest level globally
Power	
Power	Voltage 3 0-5 5v Max Current: 400mA

E. Troubleshooting

For up-to-date troubleshooting guides and frequently asked questions please visit our support website: http://www.keylok.com/support-center/overview

If after reviewing our support website your questions have still not been answered you may send an email to support@keylok.com

Or call to speak to one of our technical support representatives at 303.801.0338 x782
Mon - Thurs 7am – 4pm MST
Fri 7am – 11am MST

Index