

ROCKEY4

Developer's

Guide



Copyright © 1999-2003, Feitian Technologies Co., Ltd.
All rights reserved.
<http://www.FTsafe.com>

Software Developer's Agreement

All Products of Feitian Technologies Co., Ltd. (Feitian) including, but not limited to, evaluation copies, diskettes, CD-ROMs, hardware and documentation, and all future orders, are subject to the terms of this Agreement. If you do not agree with the terms herein, please return the evaluation package to us, postage and insurance prepaid, within seven days of their receipt, and we will reimburse you the cost of the Product, less freight and reasonable handling charges.

1. **Allowable Use** – You may merge and link the Software with other programs for the sole purpose of protecting those programs in accordance with the usage described in the Developer's Guide. You may make archival copies of the Software.
2. **Prohibited Use** – The Software or ROCKEY4 hardware dongle or any other part of the Product may not be copied, reengineered, disassembled, decompiled, revised, enhanced or otherwise modified, except as specifically allowed in item 1. You may not reverse engineer the Software or any part of the product or attempt to discover the Software's source code. You may not use the magnetic or optical media included with the Product for the purposes of transferring or storing data that was not either an original part of the Product, or a Feitian provided enhancement or upgrade to the Product.
3. **Warranty** – Feitian warrants that the ROCKEY4 dongles and Software storage media are substantially free from significant defects of workmanship or materials for a time period of twelve (12) months from the date of delivery of the Product to you.
4. **Breach of Warranty** – In the event of breach of this warranty, Feitian's sole obligation is to replace or repair, at the discretion of Feitian, any Product free of charge. Any replaced Product becomes the property of Feitian.

Warranty claims must be made in writing to Feitian during the warranty period and within fourteen (14) days after the observation of the defect. All warranty claims must be accompanied by evidence of the defect that is deemed satisfactory by Feitian. Any Products that you return to Feitian, or a Feitian authorized distributor, must be sent with freight and insurance prepaid.

EXCEPT AS STATED ABOVE, THERE IS NO OTHER WARRANTY OR REPRESENTATION OF THE PRODUCT, EXPRESSED OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE.

5. **Limitation of Feitian's Liability** – Feitian's entire liability to you or any other party for any cause whatsoever, whether in contract or in tort, including negligence, shall not exceed the price you paid for the unit of the Product that caused the damages or are the subject of, or indirectly



related to the cause of action. In no event shall Feitian be liable for any damages caused by your failure to meet your obligations, nor for any loss of data, profit or savings, or any other consequential and incidental damages, even if Feitian has been advised of the possibility of damages, or for any claim by you based on any third-party claim.

6. **Termination** – This Agreement shall terminate if you fail to comply with the terms herein. Items 2, 3, 4 and 5 shall survive any termination of this Agreement.



Contact Information

World Wide Web:

www.FTsafe.com

Feitian Technologies Co., Ltd.

Tel: +86-10-62360800, 62360900

Fax: +86-10-82070027

Email: feitian@public3.bta.net.cn

Add: 3rd Floor, Building No. 5, Jimen Hotel, Xueyuan Road, Haidian District, Beijing, PRC

Zip Code: 100088

Please Email any comments, suggestions or questions regarding this document to us at: Feitian@public3.bta.net.cn.



EC Attestation of Conformity



ROCKEY is in conformity with the protection requirements of CE Directives 89/336/EEC Amending Directive 92/31/EEC. ROCKEY satisfies the limits and verifying methods: EN55022/CISPR 22 Class B, EN55024: 1998.

FCC Standard

This device is in conformance with Part 15 of the FCC Rules and Regulation for Information Technology Equipment.

Operation of this product is subject to the following two conditions: (1) this device may not cause harmful interference, and (2) this device must accept any interference received, including interference that may cause undesired operation.

USB



This equipment is USB based.

Quick Start

- ✓ All ROCKEY dongles mentioned in this document are ROCKEY4 dongles.
- ✓ There are three types of ROCKEY4: ROCKEY4 Standard, ROCKEY4-Plus and NetROCKEY4. And every type consists of LPT mode and USB mode.
- ✓ We provide Developer's Kit (DK) to software developers for their evaluation. The DK contains everything you need to evaluate the software protection system: a Developer's Guide, a CD-ROM and a ROCKEY4 dongle(s). The ROCKEY4 dongles are exactly the same as the commercial version of the product in all respects, save one. The only difference is that the passwords burned into the demonstration dongles are publicly known (P1:C44C, P2:C8F8, P3:0799, P4:C43B). After evaluation if you decide to use the product, you can purchase dongles with secret and unique passwords, so others can not read, edit or modify the content in your dongles.
- ✓ Run *Setup.exe* under the root directory of the CD-ROM, you may finish your required installation with this installation wizard. (Refer to Chapter 3 – ROCKEY4 Development Package)
- ✓ *InstDrv.exe* under the directory *Driver* on the CD-ROM is the installation wizard for drivers, and it can also be invoked from a command line, thus it is very convenient for you to install ROCKEY drivers together with your applications on your users' computers. *InstDrv.exe* is, in fact, the package of all necessary drivers and it will install drivers according to the operating system (Refer to Chapter 3 – ROCKEY4 Development Package).
- ✓ *InstDll.dll* can install the same driver files as *InstDrv.exe*, but *InstDll.dll* does not require the Windows installation interface. Feitian has provided several programming examples for Install Shield, Vb, Delphi and Vc – each with calls to the *InstDll.dll* (Please refer to Chapter 3 – ROCKEY4 Development Package).
- ✓ After the installation of the driver, please shut down your computer and printer, and attach the LPT dongle to the parallel port of your computer, then connect the printer cable to the dongle. For USB dongles you may attach them directly to the USB port. (Note: Do not insert the USB dongles to the computer before installing the driver)
- ✓ You may find our ROCKEY tools , such as ROCKEY Editor and Envelop Encryption program, in the directory *Utility* on the CD-ROM.
- ✓ *Ryenv32.exe* is the tool for Envelop Encryption. You do not need to write any code, just choose the Win32 PEfiles, such as EXE, DLL, ARX files with your mouse, then you may encrypt them with this program. (Refer to Chapter 6 -- ROCKEY4 Envelop Encryption)

ROCKEY

- ✓ *RockEdit* was designed to help you edit, modify, test the dongle and write the content into the dongles in batch. It is the utility for you to develop encryption programs. (Refer to Chapter 5 – ROCKEY4 Editor)
- ✓ ROCKEY API is the ideal tool for you to make full use of the security functions of our dongles in your applications. You will learn how to use the dongles in no time with the reference of program samples. (Please refer to Chapter 7 – ROCKEY4 API and the programs under the directory help\beginner on the CD-ROM)
- ✓ ROCKEY ActiveX is a convenient tool for those who do not want to learn how to call API or DLL, or want to call ROCKEY functions in webpage and VBA applications.
- ✓ VBA and Vb Script users can call *Ryvc32.dll*, this DLL offers an interface for transfer character string parameter. (Please refer to the examples under the directory \Api32\Dll\Samples\VB6Str)
- ✓ Chapter 10 – Frequently Asked Questions
- ✓ Please visit our website: <http://www.FTsafe.com> We will update it frequently and you may find what you are interested in here.



Table of Contents

Chapter 1 Brief Introduction	1
1.1 About ROCKEY	1
1.2 ROCKEY4 Types.....	2
1.3 ROCKEY4 Advantages	3
1.4 How to Choose the Right Encryption Methods	4
Chapter 2 ROCKEY4 Hardware Features	5
2.1 ROCKEY4 Internal Structure	5
2.2 ROCKEY4 Hardware Interface	5
2.3 ROCKEY4 Cascading.....	5
2.4 ROCKEY4 Installation	6
Chapter 3 ROCKEY4 Development Package	7
3.1 Content of the CD-ROM	7
3.2 Installing the Development Package	8
3.3 Uninstalling the Development Package	14
3.4 Installing and Uninstalling ROCKEY4 Drivers	15
Solution 1 (Install): Install ROCKEY4 Drivers with InstDrv.exe	15
Solution 1 (Uninstall): Uninstall ROCKEY4 Drivers with InstDrv.exe	17
Solution 2 (Install): Install ROCKEY4 Drivers with InstDll.dll	20
Solution 2 (Uninstall): Uninstall ROCKEY4 Drivers with InstDll.dll	23
Chapter 4 ROCKEY4 Concepts for Developers	24
4.1 Passwords.....	24
4.2 Purchase Code	24
4.3 Hardware ID.....	24
4.4 User Data Zone	24
4.5 Module Zone	25
4.6 Module Attributes.....	25



4.7 User Algorithm Zone	25
4.8 User ID	25
4.9 Random Number	26
4.10 Seed and Return Values	26
Chapter 5 ROCKEY4 Editor	27
5.1 Brief Introduction.....	27
5.2 Operation	31
5.3 Template Storage	38
5.4 Log File and File Setting.....	38
5.5 Editor Update	39
Chapter 6 ROCKEY4 Envelope Encryption.....	41
6.1 Single File Encryption.....	42
6.1.1 Encrypt with Default Settings	42
6.1.2 Encrypt with Module	43
6.1.3 Encrypt with HID	44
6.2 Multi File Encryption	45
Chapter 7 ROCKEY4 API.....	47
7.1 ROCKEY4 Function Prototype and Definition	47
7.2 ROCKEY4 API Services	49
7.3 Return Codes.....	55
7.4 Basic Application Examples	56
7.5 Advanced Application Examples	76
Chapter 8 ROCKEY4 Hardware Algorithms	105
8.1 ROCKEY User Defined Algorithm Introduction	105
8.1.1 Instruction Format	105
8.1.2 Internal Algorithms & Application Interface	106
8.1.3 Differences between the Three Functions.....	107
8.1.4 API Interface of the User's Applications.....	108
8.2 Writing User Defined Algorithms into ROCKEY	110
8.2.1 Writing Algorithm.....	110
8.2.2 Instruction Conventions.....	110



8.3 User Defined Algorithm Examples	111
8.3.1 Basic Algorithm Application Examples	111
8.3.2 Complex Algorithm Application Examples	121
8.3.3 Advanced Algorithm Application Examples	137
8.4 Note	141
8.5 Tips	141
Chapter 9 NetROCKEY4	142
9.1 NetROCKEY4 Basic Concepts	142
9.2 NetROCKEY4 Developer's Kit	144
9.3 NetROCKEY4 Configuration Files and Tools	144
9.4 NetROCKEY4 Service Program	149
9.5 NetROCKEY4 Monitor	151
9.6 NetROCKEY4 Test Utility	155
9.7 NetROCKEY4 Envelop Encryption	160
9.8 NetROCKEY4 Function Prototype and Definition	162
9.9 NetROCKEY4 API Services	164
9.10 NetROCKEY4 Return Codes	170
9.10.1 Normal Return Codes	170
9.10.2 NetROCKEY4 Extended Return Codes	172
9.11 NetROCKEY4 Application Example	174
9.12 Quick Test	180
Chapter 10 Frequently Asked Questions	181
10.1 Trouble Shooting Suggestions	181
10.2 FAQs	181
Appendix A Content Structure of CD-ROM.....	187
Appendix B ROCKEY4 Evaluation Form.....	189
Appendix C ROCKEY4 Feedback Form.....	191

Chapter 1

Brief Introduction

1.1 About ROCKEY

ROCKEY4 is an advanced software protection system that attaches to the parallel or USB port of a computer. Your software may be duplicated, but it will only run when your ROCKEY4 “dongle” is attached to the computer. It can also limit the use of your software. Your application will interact with ROCKEY4 at start-up and during runtime. If the dongle has been removed, or if an application module has been accessed a preset number of times, it can issue an error message and terminate, or take other actions to insure compliance with your licensing agreement. ROCKEY4 is versatile enough to support other limits as well.

Unlike some competing products, ROCKEY4 is a powerful miniature computer, with a CPU, memory and specialized firmware that allow for robust interaction with your application. You may write complex algorithms that are securely stored in the dongle, and then call those algorithms from time-to-time in your application. This method for software protection is strongly recommended and is very difficult to crack. And though ROCKEY4 was designed to implement extremely high levels of security, it is also relatively easy to implement. The ROCKEY4 API set has been simplified and improved based on experience gained from earlier versions.

The ROCKEY4 product also provides an Envelope program. The ROCKEY4 Envelope (*Ryenv32.exe*) will encrypt Windows Portable Executable files (such as .dll, .exe and .arx) and is simple enough to implement in only a few seconds. The ROCKEY4 Envelope is an ideal solution if you do not possess the source code for your application, or are unfamiliar with implementing an API. A security system that combines both the API set and the Envelope program will offer the greatest level of protection.

There are three types of ROCKEY4: ROCKEY4 Standard, ROCKEY4-Plus and NetROCKEY4.

ROCKEY4 Standard is the stand alone dongle to protect the stand alone software with API encryption and Envelop encryption. ROCKEY4 Standard is comprised of CPU and memory. Developers may store & execute algorithms inside of it.

The difference between ROCKEY4 Standard and ROCKEY4 Plus is that ROCKEY4 Plus has more storage space for information and algorithms.

The NetROCKEY4 product attaches to the server and prevents oversubscription of your software. NetROCKEY4 dongle works with TCP/UDP, NetBIOS and IPX networks and is very similar in design to ROCKEY4 stand alone dongle.



There are several components to the ROCKEY4 software security solution and each of them will be discussed in this document. The following is an overview of ROCKEY4 components, along with a reference to where they will be discussed in this document:

- ✓ The ROCKEY4 Envelope program (*Ryenv32.exe*) is a fast and convenient means of encrypting .exe, .dll, .arx and other Portable Executable (PE) files. This solution is ideal if you do not have access to source code or you are not familiar with the ROCKEY4 API set. (See Chapter 6: ROCKEY4 Envelope Encryption)
- ✓ The ROCKEY4 Editor (*Rockedit.exe*) is a graphical tool for performing operations on the dongle. The Editor may be used to read data from and write data to the dongle, perform arithmetic operations in the dongle or test the dongle for malfunctions. (See Chapter 5: ROCKEY4 Editor)
- ✓ ROCKEY4 has an API set that you may use to create flexible and powerful software protection systems. This document provides VC ++ examples and other examples are provided on the CD-ROM under the Help/Beginners directory. (See Chapter 7: ROCKEY4 API)
- ✓ NetROCKEY4 includes API sample programs, NetROCKEY4 Editor, NetROCKEY4 Envelope and a network Test program. In addition, NetROCKEY4 includes a service and client Communication Configuration program to create the corresponding communication configuration files and a Monitor program that may be used to check the connectivity of the client and service programs. (See Chapter 9: NetROCKEY4)

Software Protection Mechanism of ROCKEY4: The applications must call ROCKEY hardware during run time, since the applications are dependant on the hardware and it is impossible to duplicate the chip of ROCKEY4 hardware, so it is impossible to duplicate your software, thus your software is protected from piracy.

1.2 ROCKEY4 Types

There are three types of ROCKEY4: ROCKEY4 Standard, ROCKEY4-Plus and NetROCKEY4 as discussed above. And every type consists of LPT mode and USB mode; users can choose the appropriate dongles according to their own demands. There is no difference between LPT and USB dongles of the same type when developing and using, so they can be replaced by one another without any problem.

LPT dongle is a traditional type of dongle. It attaches to the parallel port (LPT) of the computer and exchanges data with the software.

USB dongle attaches to the USB port of the computer. LPT and USB dongles of the same type are completely compatible and we provide the same access method as the LPT dongle except that the ports are different. USB dongle completely solves two problems: some servers do not have an LPT port and LPT dongles may conflict with printers. Additionally it is more convenient to attach and cascade USB dongles to computers.

The advantage of LPT dongle is that it is more compatible (almost all computers have LPT ports), and stable (LPT dongle has a longer history); the disadvantage is that the LPT port was not designed for *ROCKEY4 Developer's Guide*

connecting multiple devices. When other device (such as printer) is attached to LPT dongle, some times interference will arise (though we can make our LPT dongles compatible with above 99% LPT devices, but we can not guarantee 100%, there are always some new devices to test). Another advantage of USB dongles is that they are plug & play and much more independent; the disadvantage is that some computers do not have a USB port.

ROCKEY4 Standard and Plus were designed for protecting standalone software. Every set of software needs a dongle. NetROCKEY was designed for protecting software in LAN or WAN environments. It allows developers to limit the number of users who may attach to their application.

Comparisons between ROCKEY4 Standard, ROCKEY4 Plus and NetROCKEY4:

	ROCKEY4 Standard	ROCKEY4 Plus	NetROCKEY4
User Data Zone	24 (bytes)	120 (bytes)	120 (bytes)
Module Zone	32 (bytes)	32 (bytes)	32 (bytes)
Algorithm Zone	64 (bytes)	160(bytes)	160 (bytes)
Network Function	No	No	Yes

1.3 ROCKEY4 Advantages

1. **Compact Design** -- Feitian's parallel and USB dongles are among the smallest in the world. The parallel port dongle is 39mm x 55mm x 16mm. The USB dongle is 50mm x 17mm x 7mm.
2. **Compatible** -- ROCKEY4 LPT dongle is not only transparent to printers and scanners, but also may be cascaded (Maximum No. is 16) as long as the port voltage does not fall below 2.2V. ROCKEY4 dongles with the same type and passwords may be cascaded off the same LPT port without causing any interference.
3. **High Speed** -- ROCKEY4 was designed to process even very complex algorithms with minimal delay for your application. Users will typically notice no performance hit as a result of a ROCKEY4 implementation.
4. **Ease of Use** – Keeping the users in mind, ROCKEY4's reduced API set simplifies the programming effort and the Envelope program has also been improved with the release of ROCKEY4. The developers will get familiar with how to use the ROCKEY4 dongles in a short time, and save your time spending on software protection.
5. **High Security Levels** – Redesigned ROCKEY4 can reach a much higher security level. ROCKEY4 implements a two level security system to segregate users who need read access only from those who need administrative privileges. ROCKEY4 has a built in time gate to prevent software tracking and is powerful enough to support developer defined algorithms that can bring software protection to a new level of security.
6. **High Reliability** – Feitian employs an advanced customers managing system for ROCKEY4; we guarantee that the password of every customer is unique. And the hardware ID of every dongle is also unique. The password and hardware ID are burnt into the CPU, it is absolutely impossible to change them, even for us—the manufacturer.
7. **Broad Support for Operating Systems** -- ROCKEY4 protected applications may run on:



DOS; Windows 3.1, Windows 95/97/98/NT4.0/2000/ME/XP/2003; Linux; MAC and FreeBSD.

8. **Abundant Programming Language Interfaces** -- ROCKEY4 provides interfaces for these common development tools: TC, BC, MSC, WATCOM C, PB, TOOLBOOK, DELPHI, VFP, VB, VC, C++ BUILDER and ROCKEY4 Active X.

1.4 How to Choose the Right Encryption Methods

The security level of software protection not only depends on the dongles, but also depends on how the developers use the dongles. Even if the dongle is number one in the world, if the users do not know how to use it, its protection may be as bad as the worst dongle. ROCKEY4 dongles offer two encryption methods: envelop encryption and API encryption.

You may invoke the program *Ryenv32.exe* under the directory of *Utility* to perform the envelop encryption function. As the name indicates, envelop encryption adds an envelop to the user's files to protect the software. The envelop will call the dongle. When users execute the programs protected by envelop, the envelop program will automatically call ROCKEY4 dongle and decide whether to allow the program to continue according to the results of call. The envelop program directly encrypts the compiled files. The advantage is that it is very easy to implement and the source code does not need to be modified. The envelop method is the ideal choice if there is no time for learning the API method or if the source code is lost or unavailable. The disadvantage is that an envelop program uses a rule based encryption method, and rule based encryption methods are not as strong as methods that use an encryption key. Also, envelop encryption cannot support script languages that cannot be compiled, such as VBA.

For API encryption the developers need to choose the appropriate language interface according to their programming languages to access the dongles. API encryption was designed to be flexible; you can make full use of the encryption functions of ROCKEY4. The developers can decide where and how to encrypt their software. API encryption is more secure than envelop encryption and especially so when developers use the internal algorithm function of ROCKEY4. But API encryption must work with the original programs . It takes the developers some time to get familiarized with the API.

Chapter 2

ROCKEY4 Hardware Features

2.1 ROCKEY4 Internal Structure

At the core of ROCKEY4 LPT is an eight-pin CPU with a low working voltage of 2.2V. This CPU is in charge of the most calculation jobs. At the core of ROCKEY4 USB is a specialized CPU with a USB interface. It supports the USB 1.0 standard and is compatible with USB 2.0 standard. In addition to the CPU is a non-volatile memory chip that can save your data in the event of a power loss. The ROCKEY4 functions are divided into User, Module and Algorithm zones. The developer may store important information (such as an application serial number) inside the dongle. You can write to the ROCKEY4 dongle as many as 100,000 times – there is no appreciable limit on the numbers of reads. The ROCKEY4 chip supports special functions for random number generation, seed code generation and user defined algorithm interpretation.

2.2 ROCKEY4 Hardware Interface

ROCKEY4 LPT supports the IEEE1284 standard (ECP/EPP/SPP). The connection between the computer and ROCKEY4 is through a special communications protocol. This protocol employs an extra parity bit to insure that communications over a printer shared LPT port are not corrupted. There are also encryption functions inherent to the protocol that makes it exceedingly difficult for a rogue emulation program to spoof ROCKEY4 and break the protection mechanisms. ROCKEY4 LPT can work together with almost all kinds of printers, scanners and other LPT devices, even when two computers are directly connected by a cable in a Windows network environment.

ROCKEY4 USB supports USB Standard 1.0. At the most 16 USB dongles can attach to a computer with a USB extension HUB. The LED of ROCKEY4 USB indicates the status of the dongle. (In a normal state after the dongle is attached to the computer the LED will be on all the time. If the LED blinks it indicates that the driver is not installed. Other LED responses indicate hardware failure.)

2.3 ROCKEY4 Cascading

ROCKEY4 LPT dongles may be cascaded, or stacked on upon another on the LPT port, with no maximum limit set in software. The maximum number of dongles that may be cascaded off a single LPT port is determined by the voltage of the LPT port. The LPT port voltage is normally 3 to 5V. Each device attached to the LPT port will draw some amount of voltage, and the ROCKEY4 dongle will work properly down to a limit of 2.2V. In some environments a powered-off printer that is attached to the ROCKEY4 dongle will pull the LPT voltage below 2.2V. Feitian recommends that in this situation the printer be left powered on, or the connection to the printer be removed.

In contrast to other dongle products, ROCKEY4 dongles that have the same passwords may be cascaded. This allows multiple users to share a single printer or scanner.



cascaded off the LPT port without conflict. Cascading LPT dongles may be useful to users that need to protect multiple program modules. Also, ROCKEY USB and LPT dongles may be used together on a single PC without conflict.

2.4 ROCKEY4 Installation

The PC should be powered off before attaching the LPT dongle. The reason for this precaution is that the LPT dongle may be damaged from static electrical surges or from high voltage generated from an attached printer. However, users do commonly attach an LPT dongle to a powered on PC with no ill effect. The USB dongle was designed to be plug and play so it may always be attached to a powered on PC. The only precaution associated with the USB dongle in this regard is that it may cause stability problems with the application if the application is running when the dongle is unplugged.

Chapter 3

ROCKEY4 Development Package

You will find the program *Setup.exe* under the root directory of the CD-ROM included in the Software Developer's Kit (SDK). The contents of the CD-ROM are not zipped. Experienced developers may simply copy all necessary content to the computer.

3.1 Content of the CD-ROM

The content of the CD-ROM consists of three parts:

- ✓ Drivers under the directory *Driver*
- ✓ Tools under the directory *Utility*
- ✓ Languages developing interfaces for different operating systems

Below is the introduction of these 3 parts.

1. Drivers under the directory *Driver*

The ROCKEY4 dongles require that you install a driver that corresponds to the user's operating system. Only DOS and Win3.1 do not require a driver installation. *InstDrv.exe* under the \Driver directory is the ROCKEY4 driver installation package. This is a wizard program. It will tell you how to install the drivers step by step according to your current operating system.

You may store this program on your software CD-ROM and distribute it to your customers. *InstDrv.exe* may be invoked from a command line prompt and includes a parameter for "silent" operation. Silent operation means that the drivers will install without issuing status messages.

You may find the command line parameters of *Instdrv.exe* by clicking *help* on the installation interface (we will discuss this later).

ROCKEY4 also includes a dll library (*instdll.dll*) that you can call in your own installation program. Examples of this installation method are provided for Install Shield, VB, Delphi and Visual C. Or you may install the drivers through an external call of the *InstDrv.exe*. You may choose the appropriate driver installation method for your particular situation.

Setup.exe will automatically call *Instdrv.exe* to install the drivers. If you do not use *Setup.exe*, then you will need to call *Instdrv.exe* manually for the driver installation routine.

2. Tools under the directory *Utility*

The tools under *Utility* directory can help you to be familiar with and make full use of ROCKEY functions, there are two tools:

- Editor -- The editing and testing tool. You may edit the content of ROCKEY, test ROCKEY, test the content written into the dongle and write into the dongles in batch mode. You will find all ROCKEY functions here and it is the common tool for using ROCKEY.
- Envelop -- The envelop encryption program. It helps the users to encrypt their software without the need to write any code. It is the most convenient encryption method and can encrypt files in batch.

3. Development interfaces for different operating systems

Programming interfaces are listed under each of the API16, API32 and APIDOS directories. For Windows environments you may use the DLL interface; the DLL may be called from most Windows environments.

We also offer development package for Linux, MacOS and FreeBSD. There is also an electronic version of this developer's guide and some simple examples for API call under directory *Help* on the CD-ROM included in the SDK.

3.2 Installing the Development Package

Below we will discuss how to install and use the development package.

Step 1.

Feitian provides a *Setup.exe* installation wizard program on the CD-ROM. You may select the components you need. The installation of the drivers is also integrated in this wizard. Double click the *setup.exe* file from the root directory of the ROCKEY4 CD-ROM. You will see the first screen of the Setup Wizard pictured below. (Figure 3.1)



Figure 3.1

If you have questions or want to learn more about ROCKEY4, click the “Help” button to view the help document. If you click the “Explore CD” button Explorer will open and display the contents of the installation CD-ROM. Click “Install” or “Next” to continue. The next screen is pictured below. (See Figure 3.2)

Step 2.

Click “Prev” to return to the previous screen. Please read the license agreement carefully. If you do not agree with the terms therein, click “Exit” to quit the installation. If you agree, click “Agree”. (See Figure 3.2)

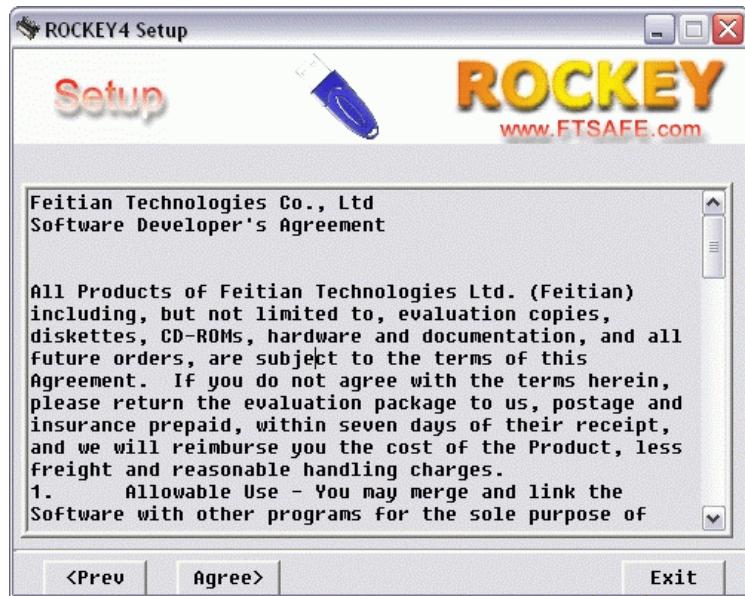


Figure 3.2

Step 3.

The default installation path for the ROCKEY4 files is "c:\Rockey" . (See Figure 3.3)



Figure 3.3



If you want to install the files to an alternate directory, select the “Browse” button. A window will display similar to the one pictured in Figure 3.4.

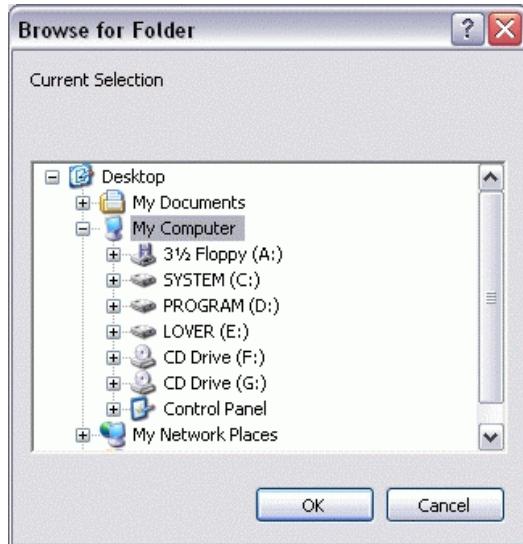


Figure 3.4

If the installation directory does not already exist, a dialog box (similar to the one pictured in Figure 3.5) will appear. Click “OK” button if you allow ROCKEY4 to create a new directory, or “Cancel” if you want to use an existing directory.



Figure 3.5

Step 4.

Figure 3.6 pictures the components you may choose to install to the previously selected directory. Be sure that you select only the check boxes for the components you want to install. If you select the “ROCKEY4 Driver” option the setup program will automatically call *InstDrv.exe* and install the drivers for mixed mode operation.



Figure 3.6

Notes: Below is a brief introduction about the components in the above figure:

ROCKEY4 Driver: Driver program for dongle applications

Net API: API interfaces for NetROCKEY4 in Windows

Win16 API: API interfaces for ROCKEY4 in Windows 3.x

Win32 API: API interfaces for ROCKEY4 in Windows 9x/2000/XP/2003

DOS Library: API interfaces for ROCKEY4 in DOS

Help: ROCKEY4 help document

Utility: ROCKEY4 tools

Step 5.

The installation program will install all components you have chosen to the specified directory. A screen similar to Figure 3.7 will appear.

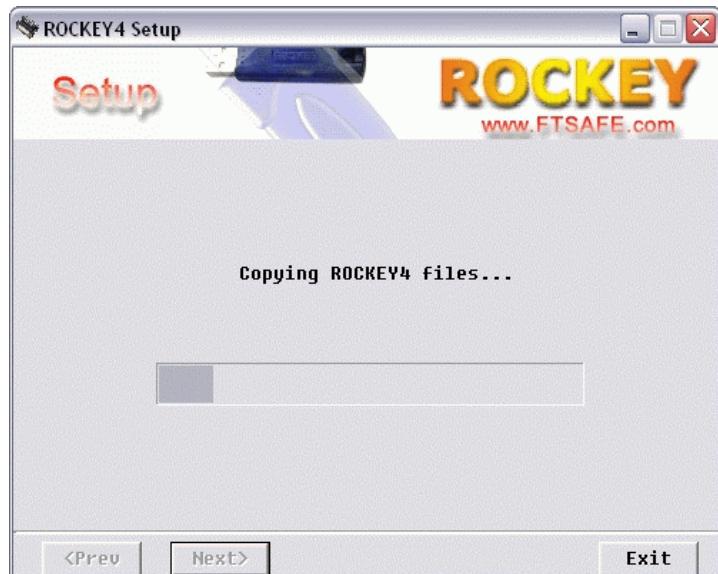


Figure 3.7

Step 6.

After the files have been copied to your computer, you will see a screen like Figure 3.8. Click the “Exit” button to complete the installation.

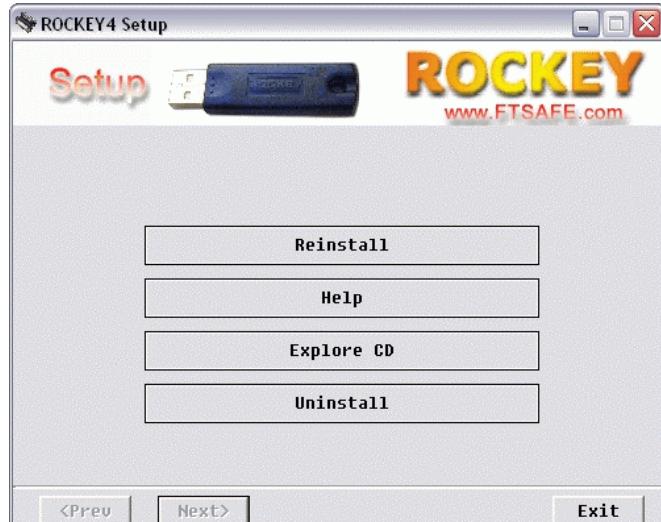


Figure 3.8

3.3 Uninstalling the Development Package

You may use Add or Remove Programs from the Windows Control Panel or select “Feitian” and then “Uninstall” from the Windows Start menu to uninstall the installed components. See Figure 3.9 and Figure 3.10.

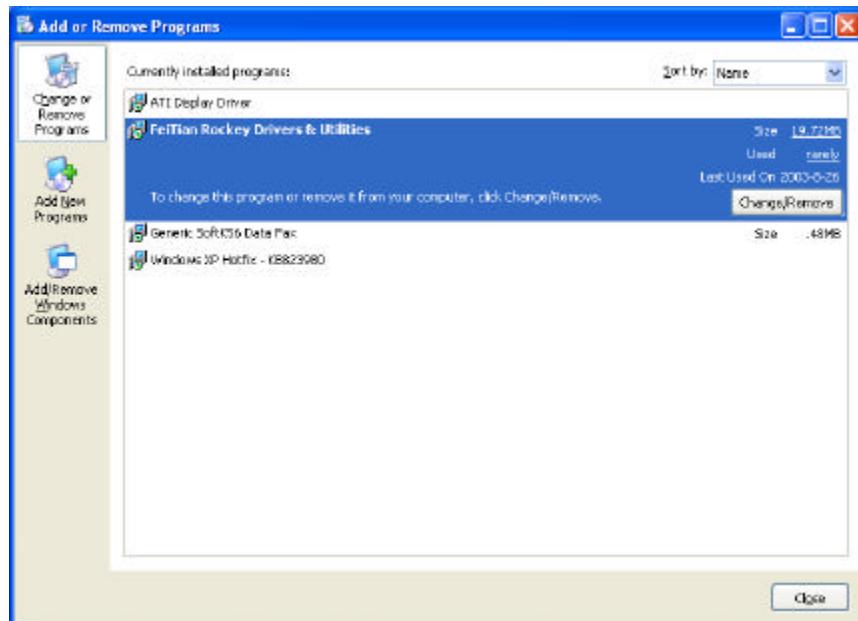


Figure 3.9



Figure 3.10

3.4 Installing and Uninstalling ROCKEY4 Drivers

Solution 1 (Install): Install ROCKEY4 Drivers with *InstDrv.exe*

InstDrv.exe is a Win32 wizard program that installs the ROCKEY driver to the computer. *Instdrv.exe* will detect the operating system for the target installation and copy only the files that are appropriate to the particular system and modify the register. This wizard can run on Chinese, English and other language platforms. It will display Simplified Chinese if Simplified Chinese is the preferred language or English for all other preferences. This program can be invoked from command line and the driver will be installed silently.



Figure 3.11

“Driver information” will display if the driver has been already installed. It is recommended to uninstall the previously installed driver first, and then install the latest driver again. In Windows 95/98 you should reboot your computer after you install or uninstall the drivers. You may also check the driver release date from this screen. You should update the driver as soon as the new driver version is released. You may find Feitian driver update information on our website (<http://www.FTsafe.com>).

Click “Install” to continue the installation. (See Figure 3.12)



Figure 3.12

You may choose and install the drivers you need here. The drivers can work in 3 modes: LPT mode, mixed mode and USB mode. LPT mode can only work with ROCKEY4 LPT. Mixed mode can work with both ROCKEY4 LPT and USB. The speed of mixed mode is a bit slower than LPT mode. USB mode can only work with ROCKEY4 USB. USB mode is faster than mixed mode but slower than LPT mode.

Speed is generally not an important factor for implementing the ROCKEY4 dongle so many customers choose to use the mixed mode operation.

“Detect-print-busy” will cause the ROCKEY4 dongle operations to pause when the printer is using the LPT port. This option will affect the operation of the dongle.

“Not-detect-print-busy” will cause the dongle to ignore the printer and may affect the print operations. In practice this is only an issue if the dongle is called during the execution of a print job.

Choose “Prev” to return to the previous Wizard program screen or “Next” to continue. After the installation is completed, next screen will appear. (Figure 3.13)

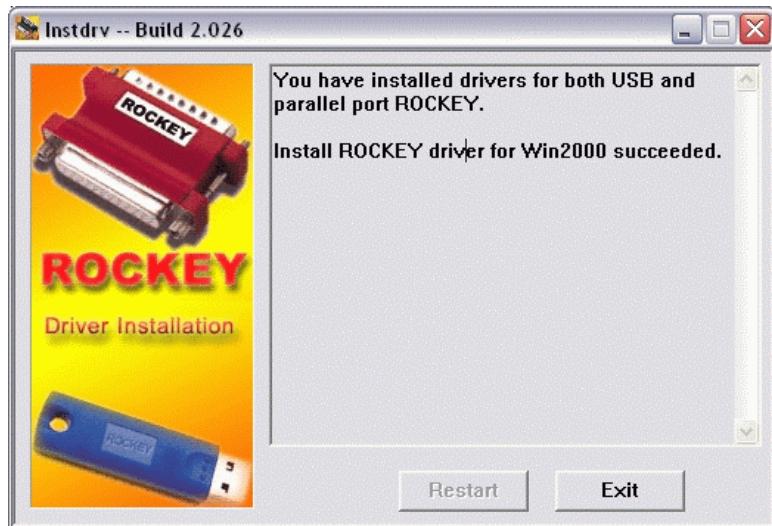


Figure 3.13

Choose “Restart” to restart your computer, “Exit” to exit the program.

Note: If the “Restart” is grayed out you do not need to restart the computer. Otherwise you will have to restart the computer.

Solution 1 (Uninstall): Uninstall ROCKEY4 Drivers with *Instdrv.exe*

Choose “Uninstall” to uninstall drivers at the first step of “The ROCKEY4 Driver Installation Wizard”. See Figure 3.14, “Driver information” is “Installed” .

ROCKEY

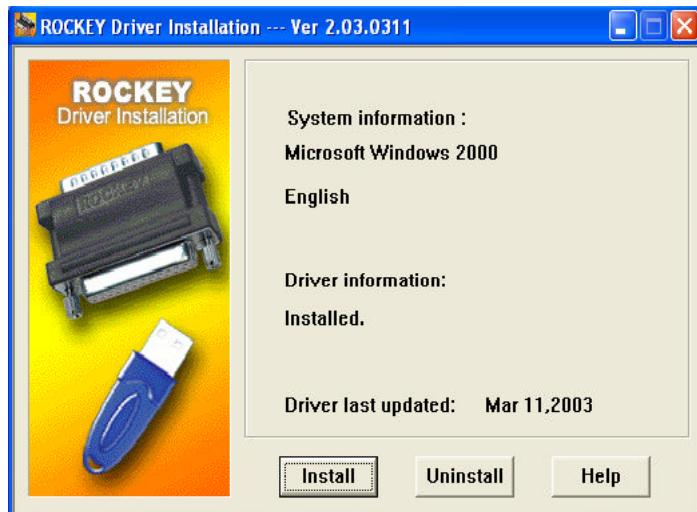


Figure 3.14

Then click “Uninstall” to remove the installed driver, see Figure 3.15.

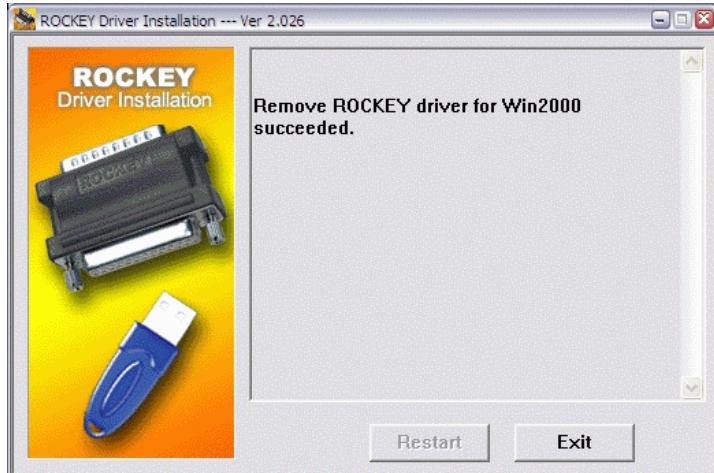


Figure 3.15

Call *InstDrv.exe* from command line

Developers may want to install the ROCKEY software and drivers on a customer’s computer without the customer seeing the installation program. In this case, the developer may call the ROCKEY driver from a command line.

From the Windows DOS prompt, or the Run dialog box, input:

instdrv /i Install the ROCKEY4 drivers without installation interface
instdrv /r Uninstall the ROCKEY4 drivers without installation interface

/s means silent mode, it will not display any message, even if the installation/uninstallation is completed or it is error message. Please see the examples below:

instdrv /i/s Install the ROCKEY4 drivers without displaying prompt messages
instdrv /r/s Uninstall the ROCKEY4 drivers without displaying prompt messages

Run *InstDrv.exe* in Windows and click the icon on the top left, the system menu will appear. Then select “About instdrv” or click “Help” button. You will get the statement of the parameters in the following window:

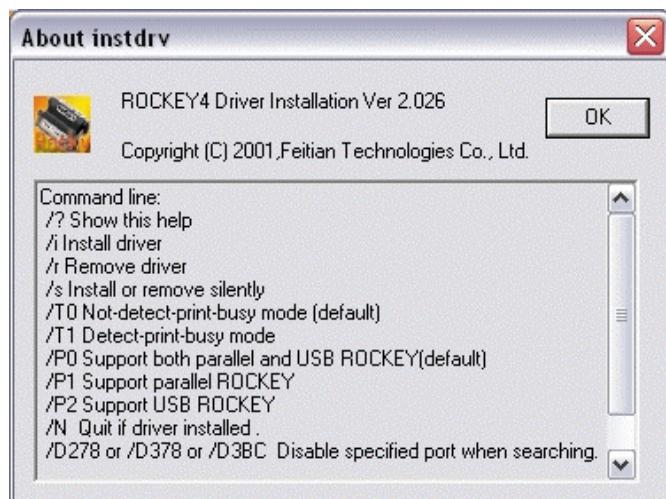


Figure 3.16

Command line parameters for *Instdrv.exe*: (not case sensitive)

/? Show command line help
/i Install driver files
/r Uninstall driver files
/s Silent install or uninstall
/T0 Not-detect-print-busy (default)
/T1 Detect-print-busy
/P0 Mixed mode, support for both parallel and USB port ROCKEY4 (default).
/P1 Support parallel port ROCKEY4 only
/P2 Support USB port ROCKEY4 only
/N Do not install a new driver if an existing driver is detected
/D278 Disable search for port number 278 (to avoid interference on some special LPT devices)

- /D378 Disable search for port number 378 (to avoid interference on some special LPT devices)
/D3BC Disable search for port number 3BC (to avoid interference on some special LPT devices)

You may select “T0” or “T1”, but not both, and you may select at most one on the /P# parameters. For example, InstDrv /I /T0 /P1 means “Install the ROCKEY4 drivers by command line, do not detect print busy, load support for parallel port ROCKEY4 only”.

If the OS is Win95/98, and you installed ROCKEY drivers before, you must restart the computer for the drivers to take effect. For Win NT/2000/XP/2003, you do not need to restart the computer. Please do not attach ROCKEY USB to the computer before you install the drivers, otherwise you have to restart your computer to make the drivers effective.

Solution 2 (Install): Install ROCKEY4 Drivers with *InstDll.dll*

InstDll.dll installs the same driver files as *InstDrv.exe*, but *InstDll.dll* is smaller because it does not require the Windows installation interface. Feitian has provided several programming examples for **Install Shield**, **VB**, **Delphi** and **VC** – each with calls to the *InstDll.dll*. Here we take **VC** and **Install Shield** as example. (Please refer to the examples and read me files under \Driver.)

VC driver installation example: (Definition and sample)

```
#pragma once

#include "windows.h"

#define INST_ERROR_SUCCESS          0    //Success
#define INST_ERROR_PARAMETER        1    //Parameter error, the parameters may conflict
                                         //with each other or neither of /i and /r is specified
#define INST_ERROR_COPYFILE         2    //Error occurs when copying files
#define INST_ERROR_NTREGISTRY       3    //Error occurs when opening Win NT/2000 registry,
                                         //please logon with administrative privilege.
#define INST_ERROR_NTSTAR           4    //Error occurs when Win NT/2000 starts device,
                                         //please reboot your computer.
#define INST_ERROR_OPENINF          5    // Error occurs when opening USB inf file
#define INST_ERROR_NONEEDINSTALL    6 //Driver is already installed, do not need to install
                                         //driver again

typedef LONG (WINAPI *PINSTROCKEY)(HWND,DWORD,char*);
typedef BOOL (WINAPI *PISNEEDRESTART)();
```

```
#include "stdafx.h"
#include "windows.h"
#include "instdll.h"

int main(int argc, char* argv[])
{
    char para[255];

    PINSTROCKEY pInstRockey=NULL;           //Installation function pointer,
                                              //please refer to install.h description
    PISNEEDRESTART pIsNeedRestart=NULL; //Judge whether it is necessary to
                                              //call restart function

    HMODULE hDll=NULL;                     //Library handle
    LONG lRet;                            //Installation return value

    //Load instdll.dll, make sure instdll.dll is under current directory or system directory
    hDll=LoadLibrary("InstDll.dll");
    if(NULL==hDll)
    {
        printf("Can't find Instdll.dll. Please copy it to this directory.\n");
        return 0;
    }

    //Get function pointer
    pInstRockey=(PINSTROCKEY)GetProcAddress(hDll,"InstRockey");
    if(NULL==pInstRockey)
    {
        printf("InstDll.dll error... Please Replace it\n");
        FreeLibrary(hDll);
        return 0;
    }
    pIsNeedRestart=(PISNEEDRESTART)GetProcAddress(hDll,"IsNeedRestart");
    if(NULL==pIsNeedRestart)
    {
        printf("Not found IsNeedRestart function, Please Replace dll with the latest version\n");
    }

    //Input installation parameter, please refer to the parameter descriptions
    printf("Please input install parameter:\n");
    /i
    gets(para);

    printf("Begin to install Rockey Driver...\n");
```

```

//Call function
lRet=pInstRockey(NULL,NULL,para);

if(INST_ERROR_SUCCESS==lRet)
{
    printf("Install Successed.\n");

    //Check if restart is needed
    if(pIsNeedRestart)
    {
        if(pIsNeedRestart())
            printf("You Have to Restart your pc to validate the driver.\n");
        else
            printf("Needn't restart your pc ^ ^\n");
    }

}

else
{
    printf("install error... error code %d\n",lRet);
}

FreeLibrary(hDll);
return 0;
}

```

Install Shield driver installation example:

```

// your global variables
INT    nValue, nResult;
STRING szValue;
STRING szDll,szRet;

program

//Insert instdll.dll to Setup Files, then use the following codes
szDll=SUPPORTDIR ^ "INSTDLL.DLL";
szValue="/i";
nResult = CallIDLLFx( szDll, "InstRockey", nValue, szValue );

if(nResult==0) then MessageBox("Setup Success",INFORMATION);

```



```
else
    Sprintf(szRet,"Setup Failed... code:%d",nResult);
    MessageBox(szRet,INFORMATION);
endif;
```

Solution 2 (Uninstall): Uninstall ROCKEY4 Drivers with *InstDll.dll*

You just need to change parameter “/i” to “ /r” in the above example to remove the ROCKEY4 drivers.
(See the example under *driver\InstDll*)

Chapter 4

ROCKEY4 Concepts for Developers

This chapter covers the basic concepts and functions of the ROCKEY4 software protection system. All ROCKEY users should read this chapter carefully to familiarize themselves with ROCKEY.

4.1 Passwords

When developers purchase ROCKEY they will get 4 16-bit passwords. The first two are Basic passwords (first grade passwords); the last two are Advanced passwords (second grade passwords). The 4 passwords for the demo dongles in the SDK are: P1: C44C, P2: C8F8, P3: 0799, P4: C43B. The passwords are “burned” into the hardware so that neither the user nor the manufacturer may change them. The developers must input the 4 passwords correctly to have full access to the dongles. Set any reference to the Advanced password set to zero in the application program that you deliver to the end user – you should never reveal the Advanced passwords to the end user in any form. The Basic passwords allow the end users to access all necessary ROCKEY functions. We will discuss when you should input the Basic passwords, and when both Basic and Advanced passwords are required, in following chapters.

4.2 Purchase Code

The Purchase Code is five to seven characters in length and corresponds to a unique customer password set. You may use the Purchase Code for reordering ROCKEY4 to be sure that all of the units in your inventory are consistent.

4.3 Hardware ID

Feitian will burn a globally unique Hardware Identification (HID) number into each ROCKEY4 dongle. The HID cannot be changed. You may use the HID to positively identify an individual ROCKEY4.

The HID is readable with the Basic passwords. It is impossible to write HID even if you have the Advanced passwords.

4.4 User Data Zone

The User Data Zone (UDZ) is a memory space that the developer can use to store data needed by the software protection system. Users can read from and write to this space at any time.

The size of the UDZ varies according to the ROCKEY4 model: for ROCKEY4 Standard the space is 24 bytes, for ROCKEY4 Plus and NetROCKEY4 the space is 120 bytes.

It may be read with Basic passwords, and may be written with Basic passwords.



4.5 Module Zone

The Module Zone was designed for multi-module encryption. It may be used to store module specific data for Envelop encryption and/or API calls.

A ROCKEY4 module is a 16-bit protected memory space. There are 16 “modules” in each ROCKEY4 dongle so as many as 16 application modules may be protected with a single ROCKEY4 dongle. The developer may write data into the ROCKEY4 modules and then use that data, along with ROCKEY4 functions, to create powerful and flexible software protection systems. If the content of the module is not “0”, you can use the module; if it is “0”, you cannot use the module. You may determine if a module is useable by analyzing the attributes of the module. The exact content can only be determined algorithmically.

ROCKEY4 modules cannot be read and it can only be written with Advanced passwords.

4.6 Module Attributes

There are two attributes associated with each ROCKEY4 module: “Zero Value” attribute and “Decrement” attribute. A 16-bit protected memory space stores an attribute of the module. The value stored in the “Zero Value” attribute indicates if the value in the associated module is “0” or not “0”. “1” indicates not “0” and that the module is usable; “0” indicates “0” and that the module is not usable. The “Decrement” attribute indicates if the value stored in the associated module can be decreased. “1” indicates it can be decreased. “0” indicates it cannot be decreased.

The “Zero Value” attribute can be read with the Basic passwords and cannot be written with Advanced passwords.

The “Decrement” attribute can be read with the Basic passwords and can be written with the Advanced passwords.

4.7 User Algorithm Zone

The User Algorithm Zone (UAZ) is a user defined area for instruction storage. The number of instructions that may be stored in the UAZ varies according to the ROCKEY4 model. The standard ROCKEY4 dongle has 32 16-bit memory units for a maximum of 32 instructions. ROCKEY4 Plus and NetROCKEY4 each supports a maximum of 80 instructions. (Please refer to Chapter 8 ROCKEY4 Hardware Algorithms.)

The User Algorithm Zone (UAZ) cannot be read and may only be written with Advanced passwords.

4.8 User ID

The User ID is a 32-bit memory allocation that may be used to store an application serial number or other identification information.

It may be read with the Basic passwords and written with the Advanced passwords.



4.9 Random Number

ROCKEY4 can generate a true random number from hardware. The random number can be used to prevent tracing or used in hardware algorithms.

4.10 Seed and Return Values

ROCKEY4 contains a proprietary algorithm that will generate four 16-bit return values from input of a 32-bit seed code and the Basic/Advanced passwords. ROCKEY dongles with the same passwords should return the same values if the seed codes are the same. The return values will be different for ROCKEY dongles with different Basic/Advanced passwords.

Chapter 5 ROCKEY4 Editor

5.1 Brief Introduction

You may use the ROCKEY4 Editor to edit data stored in ROCKEY4, test dongle functions or write in batch. The Editor is a convenient tool for learning to use ROCKEY4 and its edit operations. To run the Editor, select “*Editor*” under “*UTILITY*”. The ROCKEY4 Editor interface is organized into five parts: Tool Bar & Pull down Menu, Status Bar, Device Selector, Operation Status Log and Operation Main Window. See Figure 5.1.

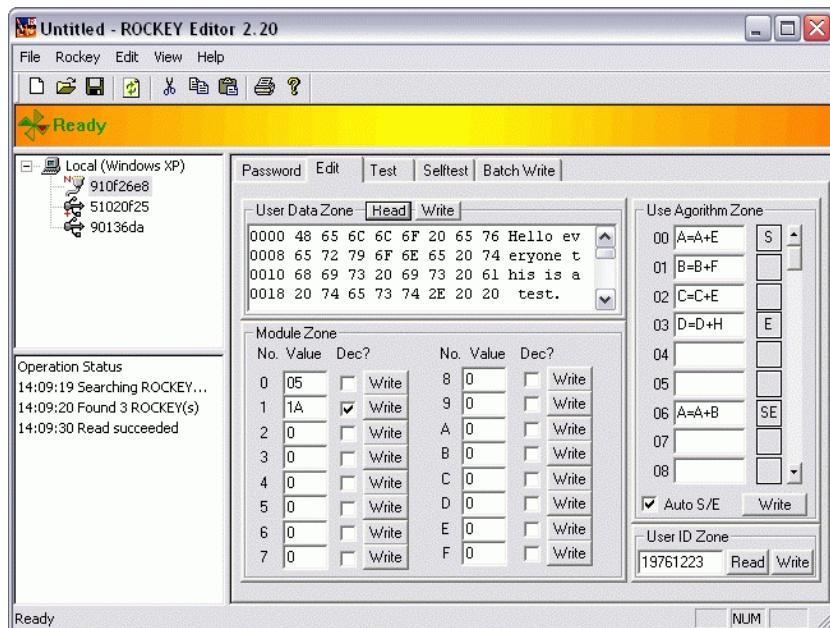


Figure 5.1

- Tool Bar & Pull down Menu** - This is the very topmost section of the screen. The typical Windows functions can be invoked from the icons or pull-down menus, such as print, save and refresh. Shortcut keys and icons are also offered. See Figure 5.2 and 5.3.

ROCKEY

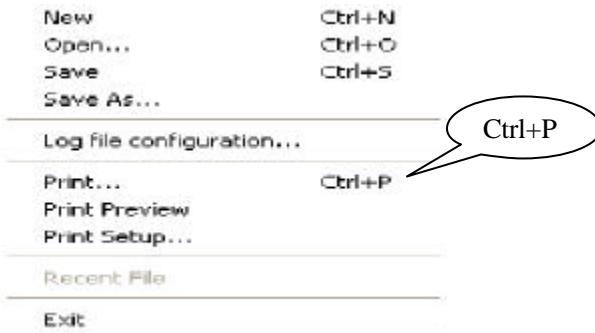


Figure 5.2

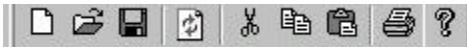


Figure 5.3

2. **Status Bar** - The Status Bar is at the bottom of the screen. The Status Bar messages are for the dongle selected in the “Device Selector” (See below) portion of the screen. Status messages are: Read, Write and Ready. See Figure 5.4.

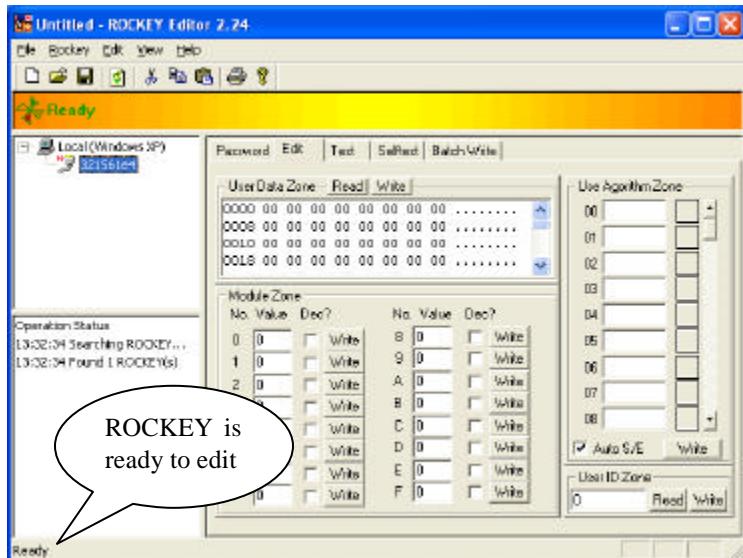


Figure 5.4

3. **Device Selector** - This is the upper left portion of the screen and shows the current OS version and ROCKEY4 dongles that are attached to the computer. Each listing will indicate the attached dongle model (ROCKEY4 Standard, ROCKEY4 Plus or NetROCKEY4) and if the dongle is a

ROCKEY

USB or Parallel device, along with its hardware ID (HID). See Figure 5.5.

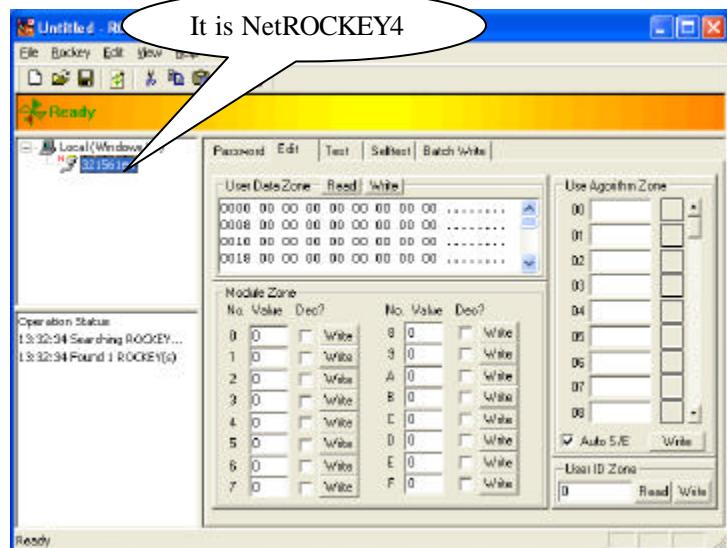


Figure 5.5

Table 5.1 lists the icons that represent the various dongle models of the ROCKEY4 software protection system. You select a dongle for editing or testing by clicking the dongle icon in this Device Selector section.

Table 5.1 ROCKEY4 Icons

ROCKEY4 Model	Icon	UDZ (Bytes)	UAZ (Instructions)
ROCKEY4-LPT		24	32
ROCKEY4-USB		24	32
NetROCKEY4-LPT		120	80
NetROCKEY4-USB		120	80

ROCKEY

ROCKEY4+-LPT		120	80
ROCKEY4+-USB		120	80

Notes:

“Device Selector” may be refreshed in two modes: Automatic refresh and Manual refresh. You may manually refresh “Device Selector” from the Rockey pull down menu and click Refresh. Or you may click the refresh button on the tool bar. If the system detects that a dongle has been reinserted, it will automatically refresh when you choose an operation. The “ Device Selector” will also refresh if another dongle is attached to the PC.

4. **Operation Status** - The time, results and error prompt of the previous operations will display here. This section is the lower left portion of the screen. See Figure 5.6.

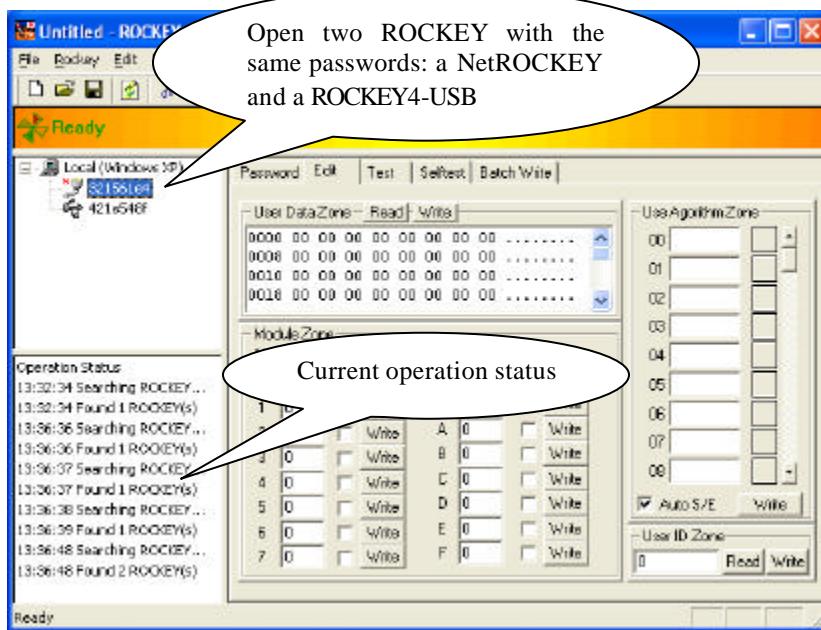


Figure 5.6

5. **Operation Main Window** - The Operation Main Window has five selection tabs: Password, Edit, Test, Self Test and Batch Write. Each tab corresponds to a screen and a function.



The Password, Self Test and Batch Write functions will affect all ROCKEY4s found attached to the computer. The Edit and Test screens will only affect the dongles selected with the Device Selector.

You may edit a template file for “Batch Write”. The template file can be saved to or imported from disk. Template files can be a convenient means of assuring that all your ROCKEY4 dongles are configured consistently.

Template files (.rki) can be opened by dragging and dropping the template file to the open Editor window. It can also be opened from the file pull down menu in the Editor or by clicking the file from Explorer. You may print preview the template file and print it out. You may use the Editor without a ROCKEY4 dongle attached to the computer and save the results to a template file. The template file can later be used for a “Write” or “Batch Write” to a ROCKEY4 dongle(s). The template file may be updated with the Editor while dongles are attached. A progress bar will display all your operation progress and you may stop your operations at any time.

Notes:

All numbers are input and displayed in hexadecimal with the exception of the number of generated seed codes in test screen.

5.2 Operation

1. Input Password

You may enter the Basic and Advanced passwords as shown in Figure 5.7.

ROCKEY

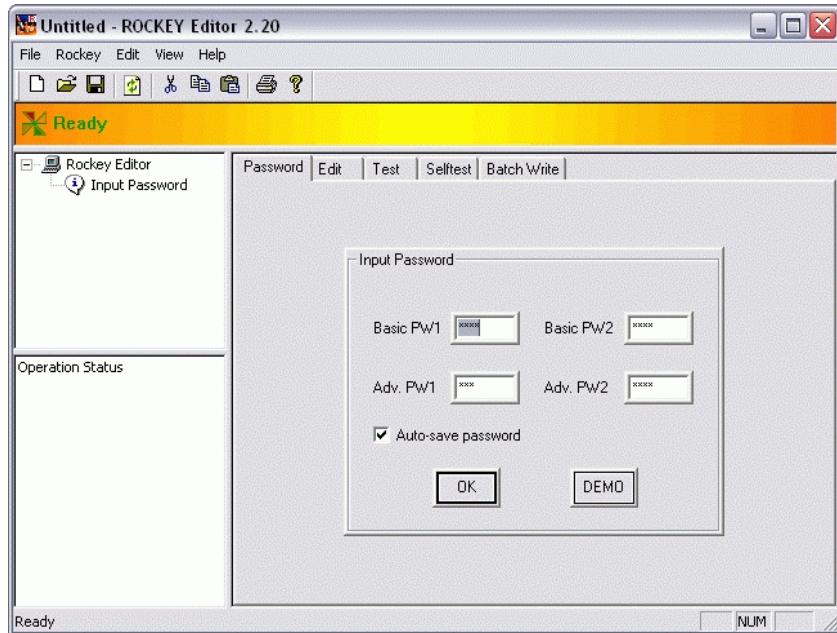


Figure 5.7

Make sure you enter the correct passwords. If the Basic passwords are incorrect Editor cannot find the dongle. If the Basic passwords are correct, and the Advanced passwords are invalid, the Editor should find the dongle and allow Read functions, but it will not allow Write functions.

If you click “DEMO” button, you may perform any operations on DEMO dongles. The 4 passwords for DEMO dongles are: P1: C44C, P2: C8F8, P3: 0799, P4: C43B.

The passwords will be saved automatically when you choose “Auto save password”. This function avoids future password entry errors.

If the entered password information corresponds with the attached ROCKEY4 dongle, selecting the “OK” or “Demo” button will take you to the “Edit” screen. The system will automatically begin to search for attached dongles.

Note:

You may edit, save, open and print template files without inputting the passwords. However, you cannot operate the dongle without at least the Basic passwords. Entering the Basic passwords will allow you to both edit template files and perform Read operations on the corresponding attached dongle.

2. Edit ROCKEY

The ROCKEY Hardware ID (HID) is displayed for all found dongles. The HID is globally unique and cannot be changed. See Figure 5.8:

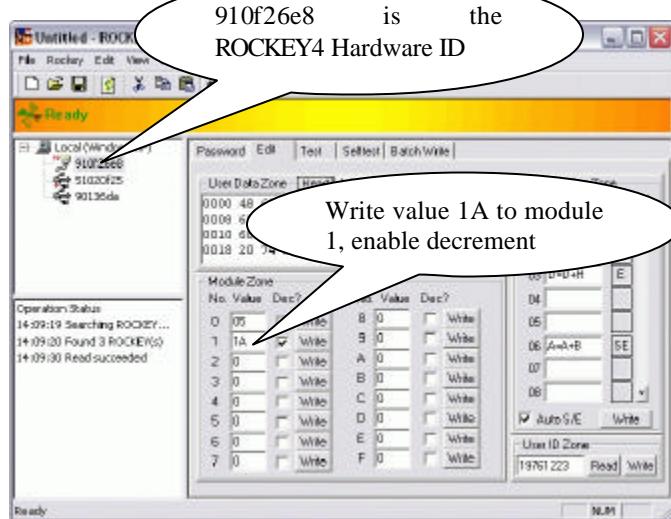


Figure 5.8

Here you may edit on the specified ROCKEY. There are four components to the Edit screen: User Data Zone (UDZ), Module Zone, User Algorithm Zone (UAZ) and the User ID Zone (UIZ).

User Data Zone (UDZ) – The UDZ is a user defined memory space. Data may be entered here in hexadecimal or ASCII text in the field provided. Press CTRL and Tab at the same time to switch between hexadecimal zone and ASCII text zone. Click the “Read” button to read data from the UDZ and “Write” to write to the UDZ. If you click Read or Write button a progress bar will appear. After the operations are finished the results will be displayed in the Operation Status section. See Figure 5.9.

Module Zone - This part of the screen is used to update the values and decrement attributes of the ROCKEY4 modules. To add new values to a module simply enter the new value in the field of the module, and click “Write”. The Decrement attribute can be likewise altered. (All 16 ROCKEY4 modules are displayed here, labeled 0 to F in hexadecimal.)

User Algorithm Zone (UAZ) – User defined algorithms may be written here. The algorithms consist of operands and operators, such as A=A+B (Please refer to Chapter 8 ROCKEY4 Hardware Algorithms). The function of the small button to the right of each instruction is to set the Start/End attribute. An algorithm will begin with an instruction marked “S” and end with one marked “E”. Single instruction algorithms will have the attribute value of “SE”. A null value attribute will be assigned to any instruction that is not explicitly the start or end of an algorithm. The “Auto S/E” option will automatically assign an attribute of “S” to the first instruction of a string and “E” to the

last instruction – and a null attribute to all attributes between the start and end of the algorithm or “SE” if there is only one instruction in the string. Click the “Write” button to write the instructions to the UAZ in the ROCKEY4 dongle. See Figure 5.9.

User ID Zone (UIZ) - User identification information may be read from or written to the UIZ of the ROCKEY4 dongle in hexadecimal. See Figure 5.9.

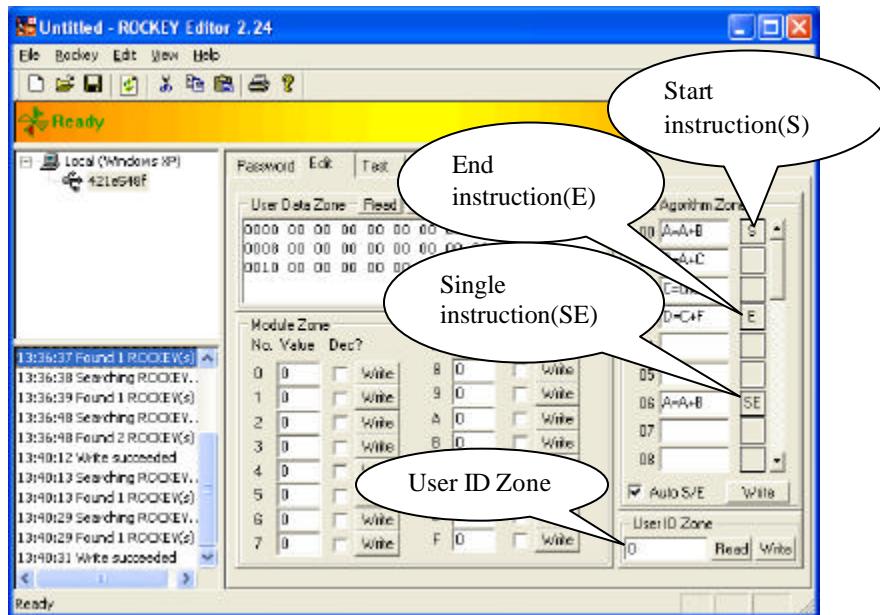


Figure 5.9

3. Test ROCKEY

There are five components to the Test screen: User Data Zone (UDZ), Calculation Zone, User ID Zone (UIZ), Module Attribute Zone and the Seed Calculation Zone. See Figure 5.10.

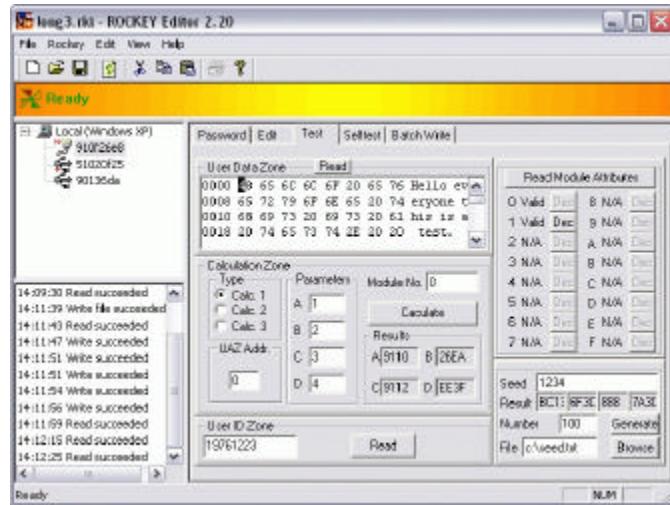


Figure 5.10

User Data Zone – The UDZ is a user defined memory space. Data may be displayed in hexadecimal form, or as ASCII text. Click the “Read” button to read data from the UDZ. You may view hexadecimal data or ASCII text here.

Calculation Zone – Be sure that you are familiar with the calculation functions before using the Calculation Zone. First select the calculation you would like to test (For Calc1 and Calc3 a “Module” entry box will appear. For Calc2 a “Seed Code” box will appear.). Then input the start address of your algorithm stored in the UAZ. The start address is where the instruction is marked with “S” or “SE”. Enter hexadecimal input values to the parameter A, B, C and D. Enter the module number or seed code and click the “Calculate” button. The results of the operation will be written to the parameters listed in the “Results” section of the Calculation Zone.

User ID Zone (UIZ) – Click the “Read” button to read the user defined ID from the UIZ of the ROCKEY4 dongle. UIZ is 32 bits in length.

Module Attribute Zone – This zone indicates the status of the Zero Value and Decrement attributes of the ROCKEY4 modules. Click the “Read Module Attributes” button to update this portion of the Test screen. “N/A” means that the Zero Value attribute is “0”. “Valid” means that the Zero Value attribute is not “0”. If the “Dec” button is grayed out the module cannot be decremented. If it can be decremented clicking the “Dec” button will reduce the value stored in the module by “1”.

Seed Calculation Zone – There are two small sections to the Seed Calculation Zone. The top section will display four calculated seed results for any entered seed code. Enter a decimal number in the “Number Generate” field in the bottom section, and that same number of random seed codes, and corresponding results, will be written to a text file defined in the “File” field. The default file is

C:\seed.txt. See Figure 5.11.

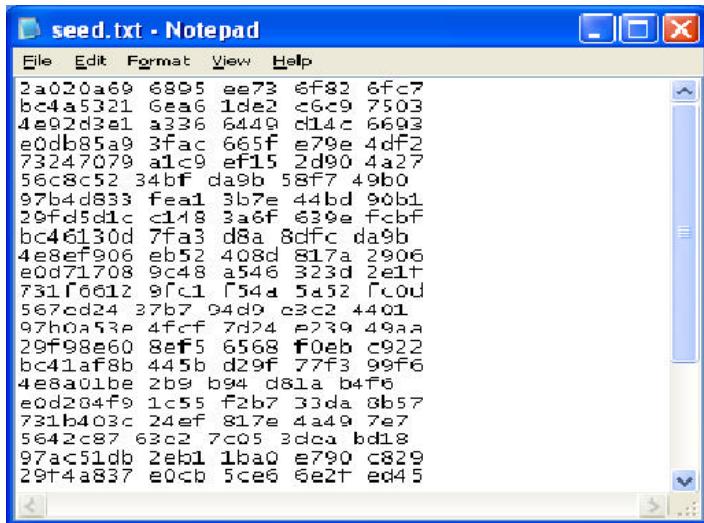


Figure 5.11

4. Self Test

Self test all ROCKETY dongles attached to the computer. See Figure 5.12.

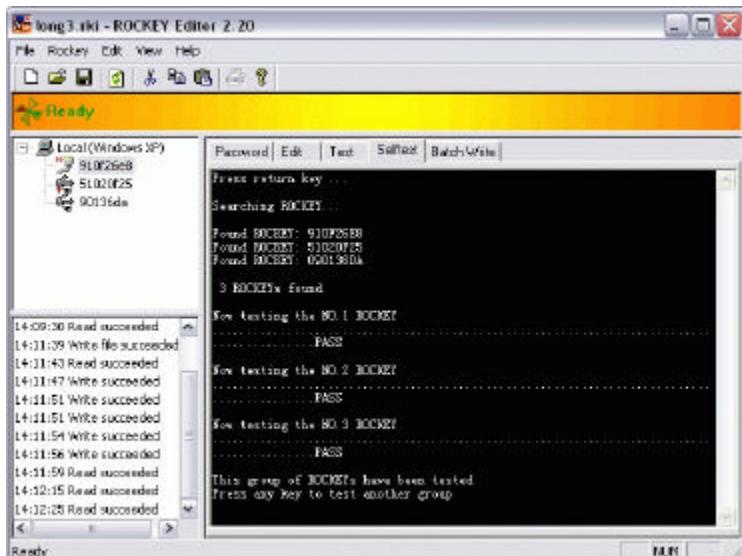


Figure 5.12

The Self Test screen will test all ROCKEY dongles automatically and write the Hardware ID (HID) to a file named “log.txt” in the current directory. (Please refer to Section Log File and File Setting)

Note:

This test is like a “Format” command in that it will delete any data or parameters stored in the dongle. It would be best to run the Self Test upon receipt of the dongle or if there is a significant problem with the dongle.

5. Batch Write

There are six components to the Batch Write screen: User Data Zone (UDZ), Module Zone, User Algorithm Zone (UAZ), User ID Zone (UIZ), Single Operation Zone and the Write All Zone. See Figure 5.13.

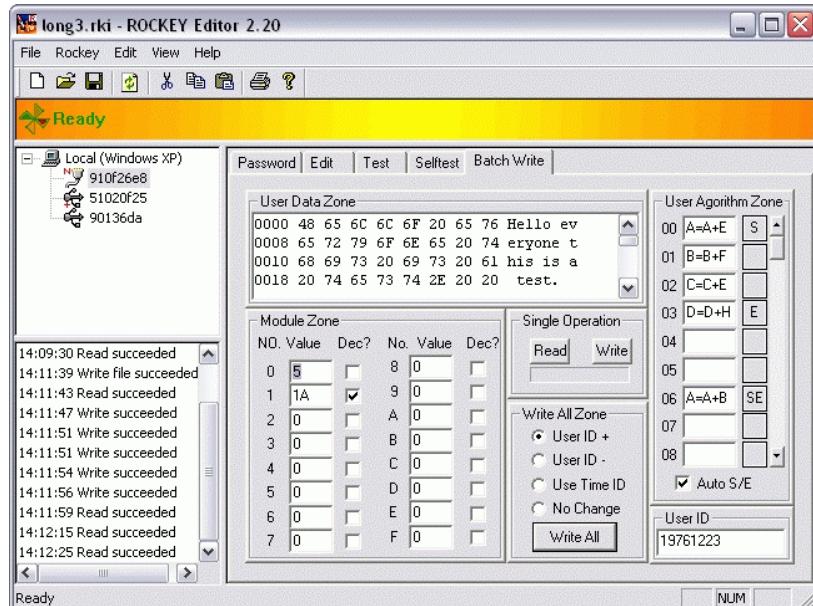


Figure 5.13

User Data Zone (UDZ) – See the previous introduction.

Module Zone – See the previous introduction.

User Algorithm Zone (UAZ) – See the previous introduction.

User ID Zone (UIZ) - See the previous introduction.

Single Operation Zone – Select the “Read” button to read the content of the ROCKEY dongle to memory. Select the Write button to write the content of memory to ROCKEY.

Write All Zone – This section will change the UIDs of all of the ROCKEY4 dongles displayed in the Device Selector section. Enter the starting UID in the “User ID” field and then click one of the “User ID” setting buttons (The setting buttons for the UID Zone are described below.) Then click “Write All” to update the attached dongles:

- **User ID +:** This button will write the value entered in the “User ID” field to the top most dongle in the Device Selector section, and increase that value by one for the subsequent dongle. It will continue to increase by “1” and write the new value to each dongle listed in the Device Selector section. For example, if the number, “123”, were entered into the User ID field and there were three dongles in the Device Selector section. Selecting “User ID +” and clicking “Write All” would write “123” to the top most dongle. The next dongle would have a UID of “124” and the last a UID of “125”.
- **User ID -:** This has the same effect as the “User ID +” field, except the UID is decreased instead of increased.
- **Use Time ID:** This button will alter all of the UIDs by a value taken from the system clock.
- **No Change:** The UID entered into the “User ID” field will be written to all of the dongles in the Device Selector section.

5.3 Template Storage

You may save the template on the disk or print out for backup. See Figure 5.14.

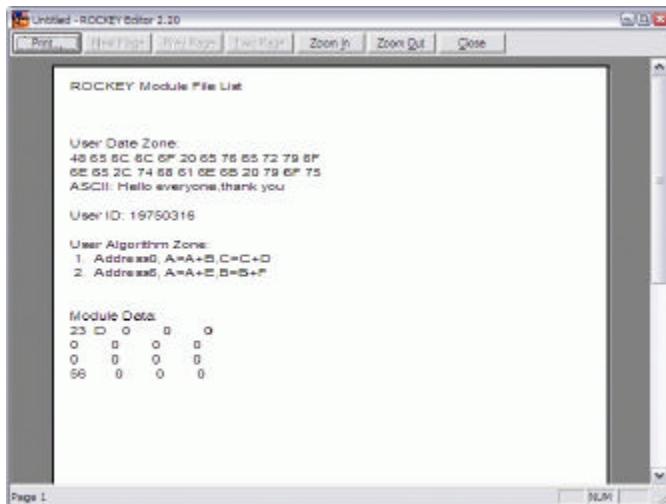


Figure 5.14

5.4 Log File and File Setting

The ROCKEY4 HID and User ID processed by the Self Test and Batch Write screens can be recorded in log file. This function is enabled by default, but you can set in the “Log file configuration” screen

that is under the “File” pull-down menu. The log function may be disabled here and the log file name altered. Please see Figure 5.15 and 5.16.

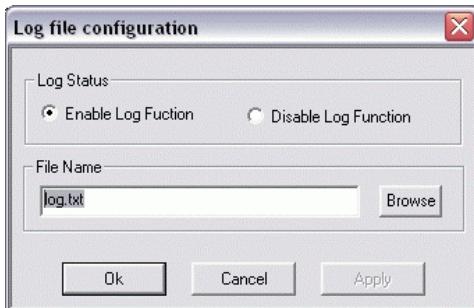


Figure 5.15

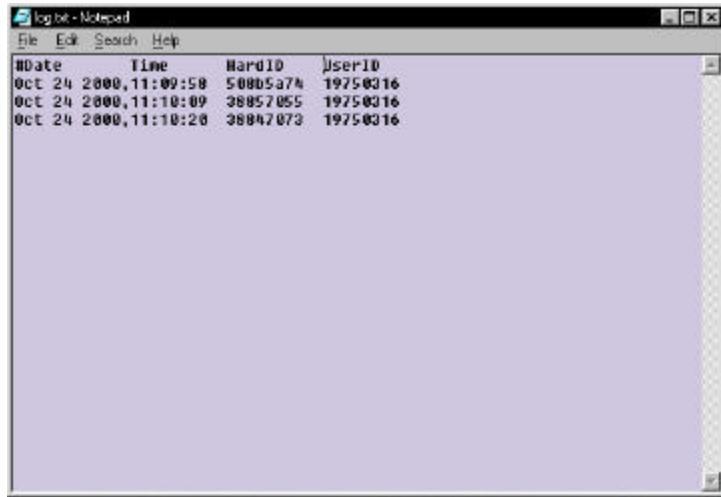


Figure 5.16

5.5 Editor Update

You may click “Help” button to check the update information of the editor. See Figure 5.17 and 5.18.



Figure 5.17

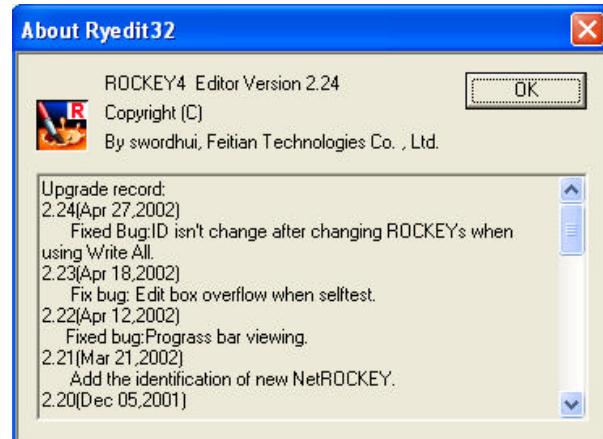


Figure 5.18

Note: If our functions can not meet all your requirements or you need some expanded functions, please contact Feitian. We will do our best to perfect it in future versions.

Chapter 6

ROCKEY4 Envelope Encryption

The ROCKEY4 Envelope program may be used for direct encryption of Win32 Portable Executable (PE) files (such as .exe, .dll or .arx). Envelope encryption is a good solution if you do not have the source code or the time to use the API functions. And Envelope encryption only works with 32-bit applications. The strongest software protection systems will use both the Envelope and API implementations.

Ryenv32.exe under directory “UTILITY” is the ROCKEY4 Envelope encryption tool. Its main interface is shown below in Figure 6.1.

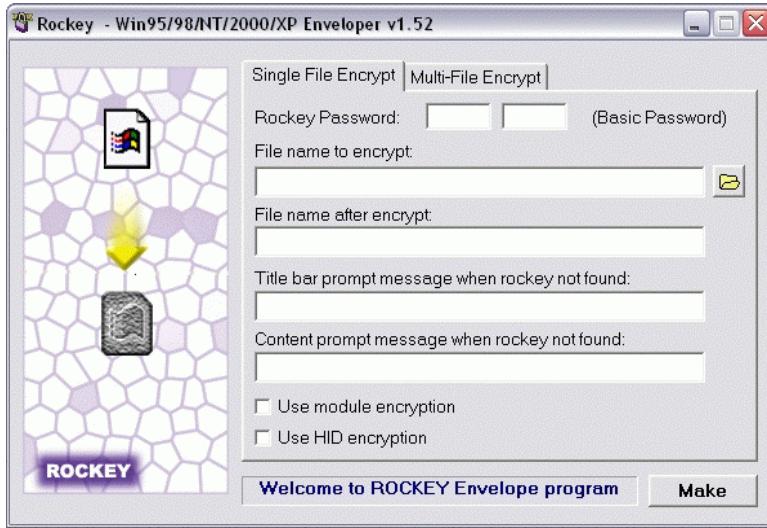


Figure 6.1

The “Single File Encrypt” program will be the most efficient protection method if only several files need to be encrypted. If many PE files need to be protected use the “Multi-File Encrypt” program.

Note: Please back up your files before you encrypt them with the Envelop program (*Ryenv32.exe*). Envelop encryption allows you to encrypt your files many times and in different ways with one ROCKEY. If you encrypt the files with the default settings, software encrypted with one ROCKEY can work with another ROCKEY with the same passwords.

If you plan to protect your software with both Envelop encryption and API encryption, please call the API first and then encrypt the software with the Envelop program.

6.1 Single File Encryption

6.1.1 Encrypt with Default Settings

The Basic passwords (the first two passwords) must be entered into the fields shown in Figure 6.2 before you can proceed. The passwords are masked with “*”. Select the file to be encrypted with “browse” button or enter its name in the “File name to encrypt:” field. Then enter the new encrypted file name and path into the “File name after encrypt:” field. The new encrypted file name is defaulted to “Encrypt.EXE”. See Figure 6.2.

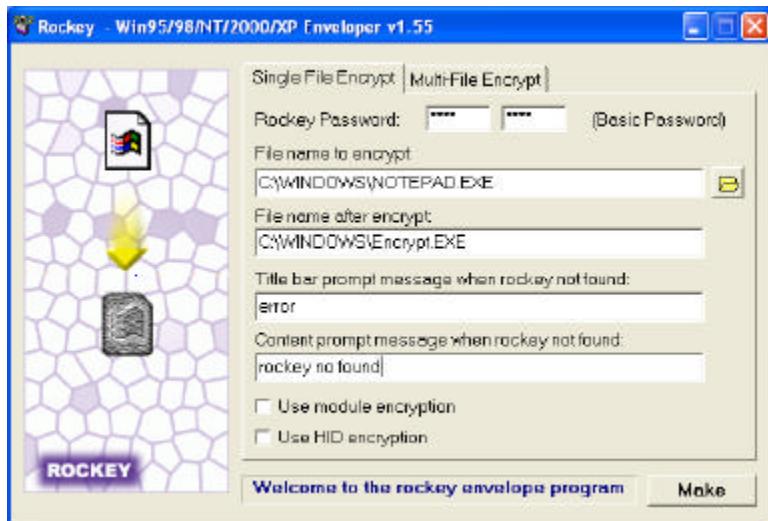


Figure 6.2

Note: The dialog box below will appear if you perform an encryption function without the correct ROCKEY4 dongle attached to the computer. When the correct ROCKEY is not found the prompt message title is “Error Message” and the prompt message content is “ROCKEY not found”.

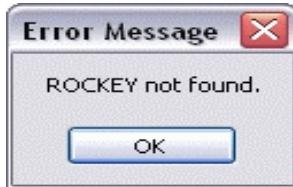


Figure 6.3

Click the “Make” button. A progress bar will appear and then the “File encryption succeeded” message will appear after a successful encryption. See Figure 6.4.



Figure 6.4

6.1.2 Encrypt with Module

If you choose “ Use module encryption”, the system will not only check whether the dongle exists and whether the passwords are valid, but also requires that the Zero Value attribute is not equal to “ 0”. To use module encryption you must write one non-zero value in the corresponding module with the Editor Program and select “ Use module encryption” from the Envelop encryption program. Be sure to enter the appropriate module number. See Figure 6.5.



Figure 6.5

6.1.3 Encrypt with HID

If you choose “Use HID encryption” the HID of the attached ROCKEY4 dongle will be copied to the Envelope program when you click the “Make” button. Thereafter, the Envelop program (and thus the application) will only run with that specific HID ROCKEY4 dongle attached to the computer. See Figure 6.6.

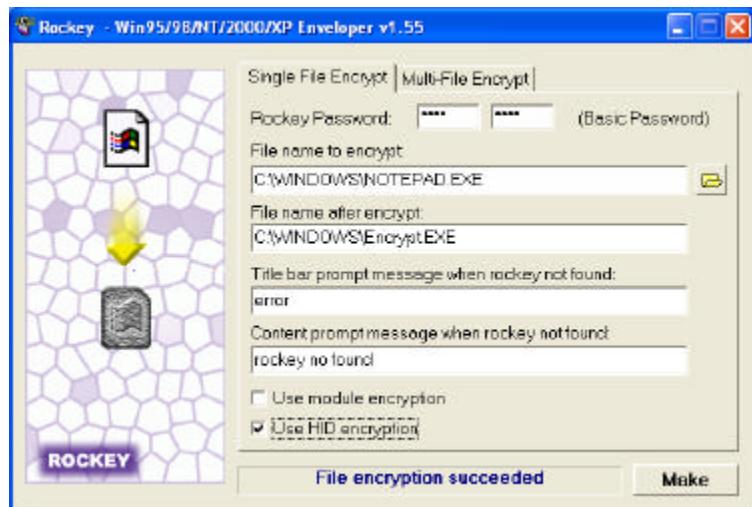


Figure 6.6

Note:

It is recommended that you use module encryption to enhance security. Combining that with HID protection will improve security still more.

Using Envelop encryption with default settings will allow end users to access two or more of your applications with a single dongle. Most developers prefer that a dongle will allow end users to access only one specific application. Use module encryption to create a one-to-one relationship between a dongle and an application. For example, use module “0” to protect application A. Use module “1” to protect application B. In this way as many as sixteen different applications may be tied to a dongle coded for only that specific application. Developers can achieve similar results by ordering dongles with password sets specified for an application. But that approach is administratively more difficult and not recommended.

6.2 Multi File Encryption

“Multi-File Encrypt” is an ideal solution when developers need to encrypt many programs or several versions of the same program. See Figure 6.7.

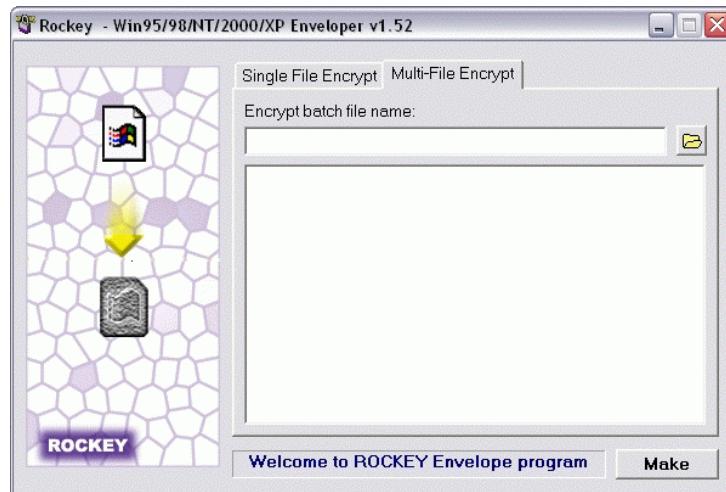


Figure 6.7

First you use a text editor to write a script file with the extension “.rbt”. There is a sample of *Ryenv32.rbt* under “UTILITY”:

```
[Common]
Password1 = c44c //ROCKEY DEMO Password 1
Password2 = c8f8 //ROCKEY DEMO Password 2
Password3 = 0
Password4 = 0
MessageTitle = "Rockey Dongle"
```

MessageContent = "The ROCKEY4 dongle is not attached or the driver is not installed."

[EncryptFile]

InputFile = c:\windows\calc.exe

OutputFile =d:\calc.exe

[EncryptFile]

InputFile = c:\windows\notepad.exe

OutputFile =d:\notepad.exe

UseHardID = YES

ModuleNo = 0

Its structure is like a Windows .INI file. There is a [Common] section we call the “general definition” section. Each file that needs to be encrypted has its own [EncryptFile] section. If a parameter is not defined in an [EncryptFile] section, its default value will come from the [Common] section as shown in the example above. Since the passwords of all files to be encrypted and the error information to be displayed are the same, they are all defined in the [Common] section. If a user needs a different prompting message, he/she may simply alter the MessageTitle and MessageContent in the [EncryptFile] section.

All definable parameters are listed below:

Password1	Password 1
Password2	Password 2
Password3	Password 3 (“ 0” , Advanced passwords not needed)
Password4	Password 4 (“ 0” , Advanced passwords not needed)
InputFile	Input file name
OutputFile	Output file name
UseHardID	Encryption with hardware ID (Default value is “NO”)
ModuleNo	Module number, checks Zero Value attribute of selected module
MessageTitle	Error message title
MessageContent	Error message body



Chapter 7 ROCKEY4 API

The ROCKEY4 Application Programming Interface (API) is the most flexible and powerful means of protecting your software. The security level of your software is determined by how you implement the API. The API set has been simplified and is intended to make the ROCKEY4 programming effort as effective as possible.

API encryption enables you to call ROCKEY in your application to enhance its security level. You may check the existence of the dongle anywhere in your application and take actions as a result of the check. You may also check the data you stored in the UDZ.

You may use the Editor program to set and write data to the modules, write algorithms to the User Algorithm Zone (UAZ), user information to the Use ID zone (UID) or take other actions. All such operations may be performed with the API.

We will take the interface of C language as example to discuss how to call the API. You may call other languages interfaces in the almost same way.

7.1 ROCKEY4 Function Prototype and Definition

WORD Rockey

(

```
    WORD    function,
    WORD*   handle,
    DWORD*  lp1,
    DWORD*  lp2,
    WORD*   p1,
    WORD*   p2,
    WORD*   p3,
    WORD*   p4,
    BYTE*   buffer
```

);

Feitian provides developers with a unified function from which they can employ all ROCKEY4 operations. This function is defined as a multi-function function.

Below is a call example for C language, and we will discuss future applications in the similar way.
retcode = Rockey(function,handle,lp1,lp2,p1,p2,p3,p4,buffer);

ROCKEY

The “ ROCKEY” function parameters are defined as:

Parameter Name	Parameter Type	Parameter Meaning
Function	A 16-bit number	API function
Handle	Address of a 16-bit number	ROCKEY4 session address
lp1	Address of a 32-bit number	long parameter 1
lp2	Address of a 32-bit number	long parameter 2
p1	Address of a 16-bit number	parameter 1
p2	Address of a 16-bit number	parameter 2
p3	Address of a 16-bit number	parameter 3
p4	Address of a 16-bit number	parameter 4
Buffer	Address of a 8-bit number	Buffer

Note: All interface parameters must be defined in your program. ROCKEY4 cannot transfer NULL or 0 pointers. Use of Null or 0 pointers in your program will generate an error.

1. “function” is a 16-bit number. It stands for the specific function and it is defined below:

RY_FIND	1	// Find ROCKEY4
RY_FIND_NEXT	2	// Find next ROCKEY4
RY_OPEN	3	// Open ROCKEY4
RY_CLOSE	4	// Close ROCKEY4
RY_READ	5	// Read ROCKEY4
RY_WRITE	6	// Write ROCKEY4
RY_RANDOM	7	// Generate Random Number
RY_SEED	8	// Generate Seed Code
RY_WRITE_USERID	9	// Write User ID
RY_READ_USERID	10	// Read User ID
RY_SET_MOUDLE	11	// Set Module
RY_CHECK_MOUDLE	12	// Check Module
RY_WRITE_ARITHMETIC	13	// Write Arithmetic
RY_CALCULATE1	14	// Calculate 1
RY_CALCULATE2	15	// Calculate 2
RY_CALCULATE3	16	// Calculate 3
RY_DECREASE	17	// Decrease Module Unit

2. “ handle” is the pointer for ROCKEY operation’s handle.

3. “lp1” and “ lp2” are the pointers for long integer parameters. Their content depends on the functions.

4. “ p1” , “ p2” , “ p3” and “ p4” are the pointers for short integer parameters. Their content depends on the functions.

5. “ buffer” is the pointer for character buffer. Its content depends on the functions.



7.2 ROCKEY4 API Services

Here we discuss the API services in detail. The following functions marked with */** require the two Advanced passwords.

Note: p3 and p4 are Advanced passwords. They are for developers to operate on the dongles. The Advanced passwords should not appear in the software you offer to your customers and you should set the two Advanced passwords “0” when searching for dongles in your application.

1. Find a ROCKEY4 dongle (RY_FIND)

Objective: To check if a specific ROCKEY4 dongle is attached to the USB or Parallel port.

Input parameters:

```
function = RY_FIND  
*p1 = Password 1  
*p2 = Password 2  
*p3 = Password 3 (optional)  
*p4 = Password 4 (optional)
```

Return value:

A return value = “0” indicates that the function worked correctly. Any other return value indicates an error. A successful operation will write the ROCKEY4 Hardware ID (HID) to *lp1.

2. Find the Next ROCKEY4 dongle (RY_FIND_NEXT)

Objective: To check if another specific ROCKEY4 dongle is attached to the USB or Parallel port.

Input parameters:

```
function = RY_FIND_NEXT  
*p1 = Password 1  
*p2 = Password 2  
*p3 = Password 3(optional)  
*p4 = Password 4(optional)  
*lp1 = The hardware ID of the last dongle found by RY_FIND or RY_FIND_NEXT
```

Return value:

A return value = “0” indicates that the function worked correctly. Any other return value indicates an error. A successful operation will write the ROCKEY4 Hardware ID (HID) to *lp1.

3. Open the ROCKEY4 dongle (RY_OPEN)



Objective: To open a ROCKEY4 dongle with specified passwords or hardware ID.

Input parameters:

```
function = RY_OPEN  
*p1 = Password 1  
*p2 = Password 2  
*p3 = Password 3(optional)  
*p4 = Password 4(optional)  
*lp1= Hardware ID
```

Return value:

A return value = “ 0” indicates that the function worked correctly. Any other return value indicates an error. A successful operation will write the handle address to the *handle parameter and write the dongle type to *lp2.

TYPE_ROCKEY4	1 //ROCKEY4 LPT
TYPE_ROCKEYP	2 //ROCKEY4-PLUS LPT
TYPE_ROCKEYUSB	3 //ROCKEY4 USB
TYPE_ROCKEYUSBP	4 //ROCKEY4-PLUS USB
TYPE_ROCKEYNET	5 //NetROCKET4 LPT
TYPE_ROCKEYUSBNET	6 //NetROCKEY4 USB

4. Close the ROCKEY4 dongle (RY_CLOSE)

Objective: To close a ROCKEY4 dongle with a specific handle.

Input parameters:

```
function = RY_CLOSE  
*handle = ROCKEY4's handle
```

Return value:

A return value = “ 0” indicates that the function worked correctly. Any other return value indicates an error.

5. Read the ROCKEY4 dongle (RY_READ)

Objective: To read the contents of the User Data Zone (UDZ).

Input parameters:

```
function = RY_READ  
*handle = ROCKEY4's handle  
*p1 = off set of UDZ(zero base)  
*p2 = length (unit is byte)  
buf = address of buffer
```

**Return value:**

A return value = “ 0” indicates that the function worked correctly. Any other return value indicates an error. A successful operation will result in the contents of the UDZ written to the memory buffer.

6. Write to the ROCKEY4 dongle (RY_WRITE)

Objective: To write data to the User Data Zone. (UDZ)

Input parameters:

function = RY_WRITE
*handle = ROCKEY4's handle
*p1 = off set of UDZ
*p2 = length (unit is byte)
buf = address of buffer

Return value:

A return value = “ 0” indicates that the function worked correctly. Any other return value indicates an error.

7. Generate a Random Number (RY_RANDOM)

Objective: To get a random number from the dongle.

Input parameters:

function = RY_RANDOM
*handle = ROCKEY4's handle

Return value:

A return value = “ 0” indicates that the function worked correctly. Any other return value indicates an error. A successful operation will result in the *p1 address populated with the random number.

8. Generate Seed Code Return Values (RY_SEED)

Objective: To get return codes from the input of a seed code.

Input parameters:

function = RY_SEED
*handle = ROCKEY4's handle
*lp2 = Seed Code

Return value:

A return value = “ 0” indicates that the function worked correctly. Any other return value indicates an error. A successful operation will result in the following addresses populated with seed code



return values:

- *p1 = Return Code 1
- *p2 = Return Code 2
- *p3 = Return Code 3
- *p4 = Return Code 4

9. Write the User ID [*] (RY_WRITE_USERID)

Objective: To write the user defined “User ID” to the User ID Zone (UIZ).

Input parameters:

- function = RY_WRITE_USERID
- *handle = ROCKEY4's handle
- *lp1 = User ID

Return value:

A return value = “0” indicates that the function worked correctly. Any other return value indicates an error.

10. Read User ID (RY_READ_USERID)

Objective: To read the user defined “User ID” from the User ID Zone (UIZ).

Input parameters:

- function = RY_READ_USERID
- *handle = ROCKEY4's handle

Return value:

A return value = “0” indicates that the function worked correctly. Any other return value indicates an error. A successful operation will result in the *lp1 address populated with the User ID.

11. Set a ROCKEY4 Module [*] (RY_SET_MOUDLE)

Objective: To write a value to a specific ROCKEY4 module and set the Decrement attribute.

Input parameters:

- function = RY_SET_MOUDLE
- *handle = ROCKEY4's handle
- *p1 = Module Number
- *p2 = Module Unit Value
- *p3 = If decreasing is allowed (1 = allowed, 0 = not allowed)

Return value:

A return value = “0” indicates that the function worked correctly. Any other return value indicates an error.



an error. A successful operation will result in module unit # “*p1” storing value “*p2” and the Decrement attribute set to “0” or “1”.

12. Check a ROCKEY4 Module (RY_CHECK_MOUDLE)

Objective: To read the attributes of a specific ROCKEY4 module.

Input parameters:

```
function = RY_CHECK_MOUDLE  
*handle = ROCKEY4's handle  
*p1 = Module Number
```

Return value:

A return value = “0” indicates that the function worked correctly. Any other return value indicates an error. A successful operation will result in “*p2” populated in the value from the Zero Value attribute (1 = module value is not zero), and “*p3” populated with the value from the Decrement attribute (1 = module can be decreased).

13. Write Arithmetic [*] (RY_WRITE_ARITHMETIC)

Objective: To write user-defined mathematical instructions to the User Algorithm Zone (UAZ).

Input parameters:

```
function = RY_WRITE_ARITHMETIC  
*handle = ROCKEY4's handle  
*p1 = position of first instruction in UAZ  
buffer = buffer address of the algorithm command string
```

Return value:

A return value = “0” indicates that the function worked correctly. Any other return value indicates an error. A successful operation will result in the UAZ populated with the algorithm command string from the buffer.

14. Calculate 1 (RY_CALCULATE1)

Objective: To return the results of a calculation performed in ROCKEY4, using input provided by the developer and the RY_CALCULATE1 function.

Input parameters:

```
function = RY_CALCULATE1  
*handle = ROCKEY4's handle  
*lp1 = Start point of calculation  
*lp2 = Module number  
*p1 = Input value 1
```

*p2 = Input value 2
*p3 = Input value 3
*p4 = Input value 4

Return value:

A return value = “ 0” indicates that the function worked correctly. Any other return value indicates an error. A successful operation will result in the addresses p1, p2, p3 and p4 populated with the results of the calculation.

15. Calculate 2 (RY_CALCULATE2)

Objective: To return the results of a calculation performed in ROCKEY4, using input provided by the developer and the RY_CALCULATE2 function.

Input parameters:

function = RY_CALCULATE2
*handle = ROCKEY4's handle
*lp1 = Start point of calculation
*lp2 = Seed Code (32-bit)
*p1 = Input value 1
*p2 = Input value 2
*p3 = Input value 3
*p4 = Input value 4

Return value:

A return value = “ 0” indicates that the function worked correctly. Any other return value indicates an error. A successful operation will result in the addresses p1, p2, p3 and p4 populated with the results of the calculation.

16. Calculate 3 (RY_CALCULATE3)

Objective: To return results of a calculation performed in ROCKEY4, using input provided by the developer and the RY_CALCULATE3 function.

Input parameters:

function = RY_CALCULATE3
*handle = ROCKEY4's handle
*lp1 = Start point of calculation
*lp2 = Module number
*p1 = Input value 1
*p2 = Input value 2
*p3 = Input value 3
*p4 = Input value 4

**Return value:**

A return value = “ 0 ” indicates that the function worked correctly. Any other return value indicates an error. A successful operation will result in the addresses p1, p2, p3 and p4 populated with the results of the calculation.

17. Decrease Module Unit (RY_DECREASE)

Objective: To decrease the value in a specified ROCKEY4 module by “ l ”.

Input parameters:

function = RY_DECREASE
 *handle = ROCKEY4's handle
 *p1 = Module number

Return value:

A return value = “ 0 ” indicates that the function worked correctly. Any other return value indicates an error. A successful operation will reduce the value stored in module *p1 by “ l ”.

7.3 Return Codes

RETURN CODE		DESCRIPTION
ERR_SUCCESS	0	Success
ERR_NO_PARALLEL_PORT	1	No parallel port on the computer
ERR_NO_DRIVER	2	No driver installed
ERR_NO_ROCKEY	3	No ROCKEY4 dongle
ERR_INVALID_PASSWORD	4	Found ROCKEY4 dongle, but Basic passwords are incorrect
ERR_INVALID_PASSWORD_OR_ID	5	Wrong password or ROCKEY4 HID
ERR_SETID	6	Set ROCKEY4 HID wrong
ERR_INVALID_ADDR_OR_SIZE	7	Read/Write address or length is wrong
ERR_UNKNOWN_COMMAND	8	No such command
ERR_NOTBELEVEL3	9	Internal error
ERR_READ	10	Read error
ERR_WRITE	11	Write error
ERR_RANDOM	12	Random number error
ERR_SEED	13	Seed code error
ERR_CALCULATE	14	Calculate error

RETURN CODE		DESCRIPTION
ERR_NO_OPEN	15	No open dongle before operating dongle
ERR_OPEN_OVERFLOW	16	Too many open dongles (>16)
ERR_NOMORE	17	No more dongle
ERR_NEED_FIND	18	No Find before FindNext
ERR_DECREASE	19	Decrease error
ERR_AR_BADCOMMAND	20	Arithmetic instruction error
ERR_AR_UNKNOWN_OPCODE	21	Arithmetic operator error
ERR_AR_WRONGBEGIN	22	Const number can't use on first arithmetic instruction
ERR_AR_WRONG_END	23	Const number can't use on last arithmetic instruction
ERR_AR_VALUEOVERFLOW	24	Const number > 63
ERR_RECEIVE_NULL	0x100	Receive null
ERR_PRNPORT_BUSY	0x101	Parallel port busy
ERR_UNKNOWN_SYSTEM	0x102	Unknown operating system
ERR_UNKNOWN	0xffff	Unknown error

7.4 Basic Application Examples

Feitian offers several program examples to help beginners quickly familiarize themselves with ROCKEY. These sample programs are intentionally simplified to illustrate various security objectives and should not be construed as sufficient for most real world implementations. These samples are for demonstration purposes only. This document is not intended to illustrate how to take full advantage of the ROCKEY software protection system – that will depend on particularities of the developer, the application and the licensing objectives. Section 7.5 Advanced Application Examples are also a good reference but the developer will need to determine the best protection method given his own constraints and objectives.

Some key points that you need to pay attention to when programming:

1. Please copy files *ryvc32.h* and *ryvc32.obj* to the appropriate directory.
2. Please add *ryvc32.obj* to your project by “Project” > “Add to project” > “Files...”
3. P3 and P4 are Advanced passwords enabling the developers to write to the dongles. They should not appear in software delivered to end users.
4. Be sure that none of the address parameters in the ROCKET4 functions are Null pointers. For example, even if you do not need Buffer, but it can not be null, otherwise the result is unpredictable.



The following sample programs are written in VC 6.0. Let us discuss how to perform the required functions step by step from an original program. Software developers who develop software in other languages please do not skip this section. There are no special developing skills for C language. Most software developers will understand the concepts illustrated here.

Note: All the original codes of sample programs are stored in *Help\Beginer* directory on the CD-ROM.

1. Original program –Step 0

This program is the original program before it is protected with ROCKEY4.

```
#include <windows.h>
#include <stdio.h>

void main()
{
// Anyone begin from here.
    printf("Hello FeiTian!\n");
}
```

2. Find a ROCKEY –Step 1

We add an operation to find the ROCKEY at the beginning of the program. If the dongle is found the program will continue. If it is not found the program will exit.

```
#include <windows.h>
#include <stdio.h>
#include "ryvc32.h"          // Include Rockey Header File

void main()
{
// =====
WORD retcode;
WORD handle, p1, p2, p3, p4;    // Rockey Variable
DWORD lp1, lp2;                // Rockey Variable
BYTE buffer[1024];             // Rockey Variable

p1 = 0xc44c;      // Rockey Demo Password 1
p2 = 0xc8f8;      // Rockey Demo Password 2
p3 = 0;           // Program needn't Password 3, Set to 0
p4 = 0;           // Program needn't Password 4, Set to 0

// Try to find specified Rockey
retcode = Rockey(RY_FIND, &handle, &lp1, &lp2, &p1, &p2, &p3, &p4, buffer);
if (retcode) // Not found
```

```
{  
    printf("ROCKEY not found!\n");  
    return;  
}  
  
// ======  
printf("Hello FeiTian!\n");  
}
```

It is a very simple security objective. We only need to call the function “Find a ROCKEY dongle”. You may refer to the introduction of the function “Find a ROCKEY dongle” in the section “ROCKEY API Services”.

For testing purposes you might try to run this program with and without the ROCKEY4 dongle attached to the computer.

3. Open the ROCKEY –Step 2

We add an operation to open ROCKEY with specified passwords at the beginning of the program. If the dongle is opened the program continue, if not the program exits.

```
#include <windows.h>  
#include <stdio.h>  
#include "ryvc32.h"      // Include Rockey Header File  
  
void main()  
{  
    // ======  
    WORD retcode;  
    WORD handle, p1, p2, p3, p4;    // Rockey Variable  
    DWORD lp1, lp2;                // Rockey Variable  
    BYTE buffer[1024];             // Rockey Variable  
  
    p1 = 0xc44c;      // Rockey Demo Password 1  
    p2 = 0xc8f8;      // Rockey Demo Password 2  
    p3 = 0;           // Program needn't Password 3, Set to 0  
    p4 = 0;           // Program needn't Password 4, Set to 0  
  
    // Try to find specified Rockey  
    retcode = Rockey(RY_FIND, &handle, &lp1, &lp2, &p1, &p2, &p3, &p4, buffer);  
    if (retcode) // Not found  
    {  
        printf("ROCKEY not found!\n");  
        return;  
    }  
  
    retcode = Rockey(RY_OPEN, &handle, &lp1, &lp2, &p1, &p2, &p3, &p4, buffer);
```

```
if (retcode) // Error
{
    printf("Error Code: %d\n", retcode);
    return;
}

// =====

printf("Hello FeiTian!\n");

retcode = Rockey(RY_CLOSE, &handle, &lp1, &lp2, &p1, &p2, &p3, &p4, buffer);
if (retcode)
{
    printf("Error Code: %d\n", retcode);
    return;
}

}
```

4. Initialize ROCKEY with Editor or API. Write “Hello FeiTian!” to the dongle and read it back from the dongle. See Step 3 and Step 4.

Initialize ROCKEY and write “Hello FeiTian!” to it – Step 3:

```
#include <windows.h>
#include <stdio.h>
#include "ryvc32.h"           // Include Rockey Header File

void main()
{
    // =====
    WORD retcode;
    WORD handle, p1, p2, p3, p4;    // Rockey Variable
    DWORD lp1, lp2;                // Rockey Variable
    BYTE buffer[1024];             // Rockey Variable

    p1 = 0xc44c;      // Rockey Demo Password 1
    p2 = 0xc8f8;      // Rockey Demo Password 2
    p3 = 0;           // Program needn't Password 3, Set to 0
    p4 = 0;           // Program needn't Password 4, Set to 0

    // Try to find Rockey
    retcode = Rockey(RY_FIND, &handle, &lp1, &lp2, &p1, &p2, &p3, &p4, buffer);
    if (retcode) // Not found
    {
        printf("ROCKEY not found!\n");
    }
}
```

```

        return;
    }

retcode = Rockey(RY_OPEN, &handle, &lp1, &lp2, &p1, &p2, &p3, &p4, buffer);
if (retcode) // Error
{
    printf("Error Code: %d\n", retcode);
    return;
}

p1 = 0;      // Pos
p2 = 14;     // Length
strcpy((char*)buffer, "Hello FeiTian! ");
retcode = Rockey(RY_WRITE, &handle, &lp1, &lp2, &p1, &p2, &p3, &p4, buffer);
if (retcode) // Error
{
    printf("Error Code: %d\n", retcode);
    return;
}
printf("Write: %s\n", buffer);

retcode = Rockey(RY_CLOSE, &handle, &lp1, &lp2, &p1, &p2, &p3, &p4, buffer);
if (retcode)
{
    printf("Error Code: %d\n", retcode);
    return;
}

}

```

In Step 3 we have written “Hello FeiTian!” to the ROCKEY dongle. In Step 4 we will read the content of the dongle.

Read dongle content – Step 4:

```

#include <windows.h>
#include <stdio.h>
#include "ryvc32.h"           // Include Rockey Header File

void main()
{
// =====
    WORD retcode;
    WORD handle, p1, p2, p3, p4; // Rockey Variable

```

```

DWORD lp1, lp2;           // Rockey Variable
BYTE buffer[1024];        // Rockey Variable

p1 = 0xc44c;    // Rockey Demo Password 1
p2 = 0xc8f8;    // Rockey Demo Password 2
p3 = 0;         // Program needn't Password 3, Set to 0
p4 = 0;         // Program needn't Password 4, Set to 0

// Try to find specified Rockey
retcode = Rockey(RY_FIND, &handle, &lp1, &lp2, &p1, &p2, &p3, &p4, buffer);
if (retcode) // Not found
{
    printf("ROCKEY not found!\n");
    return;
}

retcode = Rockey(RY_OPEN, &handle, &lp1, &lp2, &p1, &p2, &p3, &p4, buffer);
if (retcode) // Error
{
    printf("Error Code: %d\n", retcode);
    return;
}

p1 = 0;    // Pos
p2 = 14;   // Length
buffer[14] = 0;
retcode = Rockey(RY_READ, &handle, &lp1, &lp2, &p1, &p2, &p3, &p4, buffer);
if (retcode) // Error
{
    printf("Error Code: %d\n", retcode);
    return;
}

// =====
printf("%s\n", buffer);

retcode = Rockey(RY_CLOSE, &handle, &lp1, &lp2, &p1, &p2, &p3, &p4, buffer);
if (retcode)
{
    printf("Error Code: %d\n", retcode);
    return;
}
}

```

5. Generate true random number with ROCKEY – Step 5

Generate a random number when the program starts and write this random number to the dongle. The *ROCKEY4 Developer's Guide*

program should check if the random number is correct during run-time. If a sharing device is installed to this computer, and someone else runs this program also from another computer, another random number will be generated and written to the dongle. Thus the program on the first computer will be terminated since the random number is not correct.

```
#include <windows.h>
#include <stdio.h>
#include "ryvc32.h"           // Include Rockey Header File

void main()
{
    // =====
    WORD retcode;
    WORD handle, p1, p2, p3, p4;    // Rockey Variable
    DWORD lp1, lp2;                // Rockey Variable
    BYTE buffer[1024];             // Rockey Variable

    p1 = 0xc44c;      // Rockey Demo Password 1
    p2 = 0xc8f8;      // Rockey Demo Password 2
    p3 = 0;           // Program needn't Password 3, Set to 0
    p4 = 0;           // Program needn't Password 4, Set to 0

    // Try to find specified Rockey
    retcode = Rockey(RY_FIND, &handle, &lp1, &lp2, &p1, &p2, &p3, &p4, buffer);
    if (retcode) // Not found
    {
        printf("ROCKEY not found!\n");
        return;
    }

    retcode = Rockey(RY_OPEN, &handle, &lp1, &lp2, &p1, &p2, &p3, &p4, buffer);
    if (retcode) // Error
    {
        printf("Error Code: %d\n", retcode);
        return;
    }

    retcode = Rockey(RY_RANDOM, &handle, &lp1, &lp2, &p1, &p2, &p3, &p4, buffer);
    if (retcode) // Error
    {
        printf("Error Code: %d\n", retcode);
        return;
    }
    printf("Random:%04X\n", p1);
```

```

sprintf(buffer, "%04X", p1);
p1 = 0;      // Pos
p2 = 4;      // Length
retcode = Rockey(RY_WRITE, &handle, &lp1, &lp2, &p1, &p2, &p3, &p4, buffer);
if (retcode) // Error
{
    printf("Error Code: %d\n", retcode);
    return;
}
printf("Write: %s\n", buffer);

p1 = 0;      // Pos
p2 = 4;      // Length
buffer[4] = 0;
retcode = Rockey(RY_READ, &handle, &lp1, &lp2, &p1, &p2, &p3, &p4, buffer);
if (retcode) // Error
{
    printf("Error Code: %d\n", retcode);
    return;
}
printf("Read: %s\n", buffer);

if(buffer)
    printf("Hello FeiTian!\n");
else
    exit(0);

retcode = Rockey(RY_CLOSE, &handle, &lp1, &lp2, &p1, &p2, &p3, &p4, buffer);
if (retcode)
{
    printf("Error Code: %d\n", retcode);
    return;
}

}

```

6. Read the seed code return values with the Editor or API. The seed code calculation is performed inside the dongle and the algorithm is confidential. You may verify the return codes or use the return codes in an encryption routine. See Step 6 and Step 7.

Read the return codes of fixed seed code (0x12345678), Step 6:

```
#include <windows.h>
#include <stdio.h>
#include "ryvc32.h"           // Include Rockey Header File

void main()
{
    WORD retcode;
    WORD handle, p1, p2, p3, p4;    // Rockey Variable
    DWORD lp1, lp2;                // Rockey Variable
    BYTE buffer[1024];             // Rockey Variable

    p1 = 0xc44c;      // Rockey Demo Password 1
    p2 = 0xc8f8;      // Rockey Demo Password 2
    p3 = 0;           // Program needn't Password 3, Set to 0
    p4 = 0;           // Program needn't Password 4, Set to 0

    // Try to find specified Rockey
    retcode = Rockey(RY_FIND, &handle, &lp1, &lp2, &p1, &p2, &p3, &p4, buffer);
    if (retcode) // Not found
    {
        printf("ROCKEY not found!\n");
        return;
    }

    retcode = Rockey(RY_OPEN, &handle, &lp1, &lp2, &p1, &p2, &p3, &p4, buffer);
    if (retcode) // Error
    {
        printf("Error Code: %d\n", retcode);
        return;
    }

    //seed Rockey
    lp2 = 0x12345678;
    retcode = Rockey(RY_SEED, &handle, &lp1, &lp2, &p1, &p2, &p3, &p4, buffer);
    if (retcode) // Error
    {
        printf("Error Code: %d\n", retcode);
        return;
    }

    printf("Seed: %04X %04X %04X %04X\n", p1, p2, p3, p4);

    // Close Rockey
    retcode = Rockey(RY_CLOSE, &handle, &lp1, &lp2, &p1, &p2, &p3, &p4, buffer);
    if (retcode)
    {
        printf("Error Code: %d\n", retcode);
        return;
    }
```

```
    }
    printf("\n");
    getch();
}
```

Verify the return codes of the seed code to see if the program should be terminated, see Step 7:

```
#include <windows.h>
#include <stdio.h>
#include "ryvc32.h"           // Include Rockey Header File

void main()
{
    WORD retcode;
    WORD handle, p1, p2, p3, p4;    // Rockey Variable
    DWORD lp1, lp2;                // Rockey Variable
    BYTE buffer[1024];             // Rockey Variable

    p1 = 0xc44c;      // Rockey Demo Password 1
    p2 = 0xc8f8;      // Rockey Demo Password 2
    p3 = 0;           // Program needn't Password 3, Set to 0
    p4 = 0;           // Program needn't Password 4, Set to 0

    // Try to find specified Rockey
    retcode = Rockey(RY_FIND, &handle, &lp1, &lp2, &p1, &p2, &p3, &p4, buffer);
    if (retcode) // Not found
    {
        printf("ROCKEY not found!\n");
        return;
    }

    retcode = Rockey(RY_OPEN, &handle, &lp1, &lp2, &p1, &p2, &p3, &p4, buffer);
    if (retcode) // Error
    {
        printf("Error Code: %d\n", retcode);
        return;
    }

    //seed Rockey
    lp2 = 0x12345678;
    retcode = Rockey(RY_SEED, &handle, &lp1, &lp2, &p1, &p2, &p3, &p4, buffer);
    if (retcode) // Error
    {
```

```

        printf("Error Code: %d\n", retcode);
        return;
    }

    if (p1==0xD03A && p2==0x94D6 && p3==0x96A9 && p4==0x7F54)
        printf("Hello FeiTian!\n");
    else
    {
        printf("Hello error!\n");
        return;
    }

// Close Rockey
retcode = Rockey(RY_CLOSE, &handle, &lp1, &lp2, &p1, &p2, &p3, &p4, buffer);
if (retcode)
{
    printf("Error Code: %d\n", retcode);
    return;
}

}

```

7. Write User ID to the dongle with the Editor or API. User ID may be software version or product type and it may also be used in some encryption schemes. See Step 8 and Step 9.

Note: Advanced passwords are needed for Step 8.

Initialize ROCKEY and write User ID to the dongle. See Step 8:

```

#include <windows.h>
#include <stdio.h>
#include "ryvc32.h"          // Include Rockey Header File

void main()
{
    // =====
    WORD retcode;
    WORD handle, p1, p2, p3, p4;    // Rockey Variable
    DWORD lp1, lp2;                // Rockey Variable
    BYTE buffer[1024];             // Rockey Variable

    p1 = 0xc44c;      // Rockey Demo Password 1
    p2 = 0xc8f8;      // Rockey Demo Password 2
    p3 = 0x0799;      // Rockey Demo Password 3
    p4 = 0xc43b;      // Rockey Demo Password 4

```

```

// Try to find specified Rockey
retcode = Rockey(RY_FIND, &handle, &lp1, &lp2, &p1, &p2, &p3, &p4, buffer);
if (retcode) // Not found
{
    printf("ROCKEY not found!\n");
    return;
}

retcode = Rockey(RY_OPEN, &handle, &lp1, &lp2, &p1, &p2, &p3, &p4, buffer);
if (retcode) // Error
{
    printf("Error Code: %d\n", retcode);
    return;
}

lp1 = 0x88888888;
retcode = Rockey(RY_WRITE_USERID, &handle, &lp1, &lp2, &p1, &p2, &p3, &p4, buffer);
if (retcode) // Error
{
    printf("Error Code: %d\n", retcode);
    return;
}
printf("Write User ID: %08X\n", lp1);

retcode = Rockey(RY_CLOSE, &handle, &lp1, &lp2, &p1, &p2, &p3, &p4, buffer);
if (retcode)
{
    printf("Error Code: %d\n", retcode);
    return;
}

}

```

Verify the User ID. If the User ID is not 0x88888888 output “Hello DEMO!”. See Step 9:

```

#include <windows.h>
#include <stdio.h>
#include "ryvc32.h"           // Include Rockey Header File

void main()
{
    // =====
    WORD retcode;
    WORD handle, p1, p2, p3, p4; // Rockey Variable

```

```

DWORD lp1, lp2;           // Rockey Variable
BYTE buffer[1024];        // Rockey Variable

p1 = 0xc44c;    // Rockey Demo Password 1
p2 = 0xc8f8;    // Rockey Demo Password 2
p3 = 0;         // Program needn't Password 3, Set to 0
p4 = 0;         // Program needn't Password 4, Set to 0

// Try to find specified Rockey
retcode = Rockey(RY_FIND, &handle, &lp1, &lp2, &p1, &p2, &p3, &p4, buffer);
if (retcode) // Not found
{
    printf("ROCKEY not found!\n");
    return;
}

retcode = Rockey(RY_OPEN, &handle, &lp1, &lp2, &p1, &p2, &p3, &p4, buffer);
if (retcode) // Error
{
    printf("Error Code: %d\n", retcode);
    return;
}

lp1 = 0;
retcode = Rockey(RY_READ_USERID, &handle, &lp1, &lp2, &p1, &p2, &p3, &p4, buffer);
if (retcode) // Error
{
    printf("Error Code: %d\n", retcode);
    return;
}
if (lp1 == 0x88888888)
    printf("Hello FeiTian!\n");
else
{
    printf("Hello DEMO!\n");
    return;
}

retcode = Rockey(RY_CLOSE, &handle, &lp1, &lp2, &p1, &p2, &p3, &p4, buffer);
if (retcode)

{
    printf("Error Code: %d\n", retcode);
    return;
}

```

```
}
```

8. Set module value and attributes with Editor or API then check if the module is allowed to be used. Determine whether to activate associated application module. The module value may also be used by the program. Check if the module is allowed to be decreased to limit the number of software executions. See Step 10, Step 11 and Step 12.

Note: Advanced passwords are needed for Step 10.

Initialize ROCKEY and set module value. For example we set module 0 to be valid and its value cannot be decreased. See Step 10:

```
#include <windows.h>
#include <stdio.h>
#include "ryvc32.h"          // Include Rockey Header File

void main()
{
    // =====
    WORD retcode;
    WORD handle, p1, p2, p3, p4;    // Rockey Variable
    DWORD lp1, lp2;                // Rockey Variable
    BYTE buffer[1024];             // Rockey Variable

    p1 = 0xc44c;      // Rockey Demo Password 1
    p2 = 0xc8f8;      // Rockey Demo Password 2
    p3 = 0x0799;      // Rockey Demo Password 3
    p4 = 0xc43b;      // Rockey Demo Password 4

    // Try to find specified Rockey
    retcode = Rockey(RY_FIND, &handle, &lp1, &lp2, &p1, &p2, &p3, &p4, buffer);
    if (retcode) // Not found
    {
        printf("ROCKEY not found!\n");
        return;
    }

    retcode = Rockey(RY_OPEN, &handle, &lp1, &lp2, &p1, &p2, &p3, &p4, buffer);
    if (retcode) // Error
    {
        printf("Error Code: %d\n", retcode);
        return;
    }
}
```

```
}
```

p1 = 0;
p2 = 3;
p3 = 0;
retcode = Rockey(RY_SET_MOUDLE, &handle, &lp1, &lp2, &p1, &p2, &p3, &p4, buffer);
if (retcode)
{
 printf("Error Code: %d\n", retcode);
 return;
}
printf("Set Moudle 0: Pass = %04X Decrease no allow\n", p2);

retcode = Rockey(RY_CLOSE, &handle, &lp1, &lp2, &p1, &p2, &p3, &p4, buffer);
if (retcode)
{
 printf("Error Code: %d\n", retcode);
 return;
}

```
}
```

If module 0 is valid in the program, output “Hello Feitian!”. Otherwise terminate or exit the program.
See Step 11:

```
#include <windows.h>  
#include <stdio.h>  
#include "ryvc32.h"           // Include Rockey Header File  
  
void main()  
{  
    // ======  
    WORD retcode;  
    WORD handle, p1, p2, p3, p4;    // Rockey Variable  
    DWORD lp1, lp2;                // Rockey Variable  
    BYTE buffer[1024];             // Rockey Variable  
  
    p1 = 0xc44c;      // Rockey Demo Password 1  
    p2 = 0xc8f8;      // Rockey Demo Password 2  
    p3 = 0;          // Program needn't Password 3, Set to 0  
    p4 = 0;          // Program needn't Password 4, Set to 0  
  
    // Try to find specified Rockey  
    retcode = Rockey(RY_FIND, &handle, &lp1, &lp2, &p1, &p2, &p3, &p4, buffer);
```

```
if (retcode) // Not found
{
    printf("ROCKEY not found!\n");
    return;
}

retcode = Rockey(RY_OPEN, &handle, &lp1, &lp2, &p1, &p2, &p3, &p4, buffer);
if (retcode) // Error
{
    printf("Error Code: %d\n", retcode);
    return;
}

p1 = 0;
retcode = Rockey(RY_CHECK_MOUDLE, &handle, &lp1, &lp2, &p1, &p2, &p3, &p4, buffer);
if (retcode)
{
    printf("Error Code: %d\n", retcode);
    return;
}

if (p2)
    printf("Hello FeiTian!\n");
else
    return;

retcode = Rockey(RY_CLOSE, &handle, &lp1, &lp2, &p1, &p2, &p3, &p4, buffer);
if (retcode)

{
    printf("Error Code: %d\n", retcode);
    return;
}

}
```

In Step 10 we set p2=3(allowed software run times) and p3=1-Decrement allowed). That is to say module 0(p1=0) sets the maximum software run time to 3. See Step 12:

```
#include <windows.h>
#include <stdio.h>
#include "ryvc32.h"           // Include Rockey Header File

void main()
{
```

```

// =====
WORD retcode;
WORD handle, p1, p2, p3, p4; // Rockey Variable
DWORD lp1, lp2; // Rockey Variable
BYTE buffer[1024]; // Rockey Variable

p1 = 0xc44c; // Rockey Demo Password 1
p2 = 0xc8f8; // Rockey Demo Password 2
p3 = 0; // Program needn't Password 3, Set to 0
p4 = 0; // Program needn't Password 4, Set to 0

// Try to find specified Rockey
retcode = Rockey(RY_FIND, &handle, &lp1, &lp2, &p1, &p2, &p3, &p4, buffer);
if (retcode) // Not found
{
    printf("ROCKEY not found!\n");
    return;
}

retcode = Rockey(RY_OPEN, &handle, &lp1, &lp2, &p1, &p2, &p3, &p4, buffer);
if (retcode) // Error
{
    printf("Error Code: %d\n", retcode);
    return;
}

p1 = 0;
retcode = Rockey(RY_CHECK_MOUDLE, &handle, &lp1, &lp2, &p1, &p2, &p3, &p4, buffer);
if (retcode)
{
    printf("Error Code: %d\n", retcode);
    return;
}

if (p2!=1)
{
    printf("Update Please!\n");
    return;
}

if(p3==1)
{
    p1=0;
    retcode = Rockey(RY_DECREASE, &handle, &lp1, &lp2, &p1, &p2, &p3, &p4, buffer);
    if(retcode)
    {
}
}

```

```

        printf("Error Code: %d\n", retcode);
        return;
    }

}

// =====

printf("Hello FeiTian!\n");

retcode = Rockey(RY_CLOSE, &handle, &lp1, &lp2, &p1, &p2, &p3, &p4, buffer);
if (retcode)

{
    printf("Error Code: %d\n", retcode);
    return;
}

}

```

9. Multi ROCKEY dongles with the same passwords may work on the same computer no matter whether the dongle types are the same or not. The program can distinguish the dongles because every dongle has a unique hardware ID. See Step 13:

```

#include <windows.h>
#include <stdio.h>
#include "ryvc32.h"           // Include Rockey Header File

void main()
{
int i, rynum;
WORD retcode;
WORD handle[16], p1, p2, p3, p4;      // Rockey Variable
DWORD lp1, lp2;                      // Rockey Variable
BYTE buffer[1024];                   // Rockey Variable

p1 = 0xc44c;             // Rockey Demo Password 1
p2 = 0xc8f8;             // Rockey Demo Password 2
p3 = 0;                  // Program needn't Password 3, Set to 0
p4 = 0;                  // Program needn't Password 4, Set to 0

// Try to find all Rockey
for (i=0;i<16;i++)
{
    if (0 == i)
    {
        retcode = Rockey(RY_FIND, &handle[i], &lp1, &lp2, &p1, &p2, &p3, &p4, buffer);
    }
}

```

```
if (retcode == ERR_NOMORE)
    break;
}
else
{
// Notice : lp1 = Last found hardID
retcode = Rockey(RY_FIND_NEXT, &handle[i], &lp1, &lp2, &p1, &p2, &p3, &p4, buffer);
    if (retcode == ERR_NOMORE)
        break;
}

if (retcode) // Error
{
    printf("Error Code: %d\n", retcode);
    return;
}

printf("Found Rockey: %08X ", lp1);

retcode = Rockey(RY_OPEN, &handle[i], &lp1, &lp2, &p1, &p2, &p3, &p4, buffer);
if (retcode) // Error
{
    printf("Error Code: %d\n", retcode);
    return;
}

printf("(");
switch(lp2)
{
    case TYPE_ROCKEY4:
        printf("Rockey 4 Standard Parallel Port");
        break;
    case TYPE_ROCKEY4P:
        printf("Rockey 4 Plus Parallel Port");
        break;
    case TYPE_ROCKEYUSB:
        printf("Rockey 4 Standard USB Port");
        break;
    case TYPE_ROCKEYUSBP:
        printf("Rockey 4 Plus USB Port");
        break;
    case TYPE_ROCKEYNET:
        printf("Rockey 4 Net Parallel Port");
        break;
    case TYPE_ROCKEYUSBNET:
        printf("Rockey 4 Net USB Port");
        break;
    default:
}
```

```

        printf("Unknown Type");
    }
    printf("\n");
}
printf("\n");

rnum = i;

// Do our work
for (i=0;i<rnum;i++)
{
    // Read Rockey user memory
    p1 = 0;      // Pos
    p2 = 12;     // Length
    buffer[12] = 0;
    retcode = Rockey(RY_READ, &handle[i], &lp1, &lp2, &p1, &p2, &p3, &p4, buffer);
    if (retcode) // Error
    {
        printf("Error Code: %d\n", retcode);
        return;
    }
    printf("%s\n", buffer);      // Output

    lp1=0;
    retcode = Rockey(RY_READ_USERID, &handle[i], &lp1, &lp2, &p1, &p2, &p3, &p4,
buffer);
    if (retcode) // Error
    {
        printf("Error Code: %d\n", retcode);
        return;
    }
    printf("Read User ID: %08X\n", lp1);

    p1=0;
    retcode = Rockey(RY_CHECK_MOUDLE, &handle[i], &lp1, &lp2, &p1, &p2, &p3, &p4,
buffer);
    if (retcode) // Error
    {
        printf("Error Code: %d\n", retcode);
        return;
    }

    printf("Check Moudle 0: ");
    if (p2)
        printf("Allow   ");
    else
        printf("No Allow  ");
}

```

```
if (p3)
    printf("Allow Decrease\n");
else
    printf("Not Allow Decrease\n");

}

// Close all opened Rockey
for (i=0;i<rynum;i++)
{
    retcode = Rockey(RY_CLOSE, &handle[i], &lp1, &lp2, &p1, &p2, &p3, &p4, buffer);
    if (retcode)
    {
        printf("Error Code: %d\n", retcode);
        return;
    }
}

}
```

At most 16 dongles may be attached to the same computer at the same time. The program can access any dongle you specify.

In the above program we defined a handle array to save the opened ROCKEY handle to prepare for the next operation on the specified dongle. When we find the dongle we open it and we close all opened ROCKEY handles before exiting the program. Developers had better operate in this way but for a large program it is OK to open/close the dongle just once at the beginning/end of the program. Frequent open and close operations will reduce performance. We open the dongle in share mode so that another program may also simultaneously operate on the dongle.

Note: We called function RY_OPEN and RY_CLOSE in the above program. We must open ROCKEY before all operations with the exceptions of RY_FIND and RY_FIND_NEXT. This is similar to the operation on the disk files. You should close the dongle immediately after finishing dongle related operations.

7.5 Advanced Application Examples

This section is dedicated to providing additional illustrative examples of methods you may employ to protect your software with ROCKEY4. These examples are intentionally simplified and not intended to be complete solutions for software protection. The method appropriate for your application will depend on constraints set by your licensing agreement and other factors. (If you are familiar with the API call already, you may skip to Chapter 8 ROCKEY4 Hardware Algorithms.)



1. User Data Zone advanced application

In Step 14 we will write “Hello FeiTian!” to User Data Zone (UDZ). In general we would write “Hello FeiTian!” to the UDZ as one character string, but security may be enhanced by writing it in two parts and then later combining the character strings.

```
#include <windows.h>
#include <stdio.h>
#include <conio.h>
#include "ryvc32.h"

void ShowERR(WORD retcode)
{
    if (retcode == 0) return;
    printf("Error Code: %d\n", retcode);
}

void main()
{
    WORD handle[16], p1, p2, p3, p4, retcode;
    DWORD lp1, lp2;
    BYTE buffer[1024];
    BYTE buf[1024];

    int i, j;

    p1 = 0xc44c;
    p2 = 0xc8f8;
    p3 = 0;
    p4 = 0;

    retcode = Rockey(RY_FIND, &handle[0], &lp1, &lp2, &p1, &p2, &p3, &p4, buffer);
    if (retcode)
    {
        ShowERR(retcode);
        return;
    }
    printf("Find Rock: %08X\n", lp1);

    retcode = Rockey(RY_OPEN, &handle[0], &lp1, &lp2, &p1, &p2, &p3, &p4, buffer);
    if (retcode)
    {
        ShowERR(retcode);
        return;
    }
```

```
i = 1;
while (retcode == 0)
{
    retcode = Rockey(RY_FIND_NEXT, &handle[i], &lp1, &lp2, &p1, &p2, &p3, &p4, buffer);
    if (retcode == ERR_NOMORE) break;
    if (retcode)
    {
        ShowERR(retcode);
        return;
    }

    retcode = Rockey(RY_OPEN, &handle[i], &lp1, &lp2, &p1, &p2, &p3, &p4, buffer);
    if (retcode)
    {
        ShowERR(retcode);
        return;
    }

    i++;
    printf("Find Rock: %08X\n", lp1);
}
printf("\n");

for (j=0;j<i;j++)
{
    p1 = 0;
    p2 = 10;
    strcpy((char*)buffer, "Hello ");
    retcode = Rockey(RY_WRITE, &handle[j], &lp1, &lp2, &p1, &p2, &p3, &p4, buffer);
    if (retcode)
    {

        ShowERR(retcode);
        return;
    }
    printf("Write: Hello \n");

    p1 = 12;
    p2 = 12;
    strcpy((char*)buffer, "FeiTian!");
    retcode = Rockey(RY_WRITE, &handle[j], &lp1, &lp2, &p1, &p2, &p3, &p4, buffer);
    if (retcode)
    {
        ShowERR(retcode);
        return;
    }
    printf("Write: FeiTian!\n");
}
```

```
p1 = 0;
p2 = 10;
memset(buffer, 0, 64);
retcode = Rockey(RY_READ, &handle[j], &lp1, &lp2, &p1, &p2, &p3, &p4, buffer);
if (retcode)
{
    ShowERR(retcode);
    return;
}
printf("Read: %s\n", buffer);

p1 = 12;
p2 = 12;
memset(buf, 0, 64);
retcode = Rockey(RY_READ, &handle[j], &lp1, &lp2, &p1, &p2, &p3, &p4, buf);
if (retcode)
{
    ShowERR(retcode);
    return;
}
printf("Read: %s\n", buf);

printf("\n");
printf("%s\n", strcat(buffer,buf));

retcode = Rockey(RY_CLOSE, &handle[j], &lp1, &lp2, &p1, &p2, &p3, &p4, buffer);
if (retcode)
{
    ShowERR(retcode);
    return;
}
getch();
}

}
```

Step 15: You may write a serial number in the User Data Zone (UDZ) and then verify it during run time as a means of protecting and controlling a program module.

```
#include <windows.h>
#include <stdio.h>
#include <conio.h>
#include "ryvc32.h"

void ShowERR(WORD retcode)
{
    if (retcode == 0) return;
    printf("Error Code: %d\n", retcode);
}

void main()
{
    WORD handle[16], p1, p2, p3, p4, retcode;
    DWORD lp1, lp2;
    BYTE buffer[1024];

    int i, j;

    p1 = 0xc44c;
    p2 = 0xc8f8;
    p3 = 0x0799;
    p4 = 0xc43b;

    retcode = Rockey(RY_FIND, &handle[0], &lp1, &lp2, &p1, &p2, &p3, &p4, buffer);
    if (retcode)
    {
        ShowERR(retcode);
        return;
    }
    printf("Find Rock: %08X\n", lp1);

    retcode = Rockey(RY_OPEN, &handle[0], &lp1, &lp2, &p1, &p2, &p3, &p4, buffer);
    if (retcode)
    {
        ShowERR(retcode);
        return;
    }

    i = 1;
    while (retcode == 0)
    {
        retcode = Rockey(RY_FIND_NEXT, &handle[i], &lp1, &lp2, &p1, &p2, &p3, &p4, buffer);
        if (retcode == ERR_NOMORE) break;
        if (retcode)
        {
            ShowERR(retcode);
        }
    }
}
```



```
        return;
    }

    retcode = Rockey(RY_OPEN, &handle[i], &lp1, &lp2, &p1, &p2, &p3, &p4, buffer);
    if (retcode)
    {
        ShowERR(retcode);
        return;
    }

    i++;
    printf("Find Rock: %08X\n", lp1);
}
printf("\n");

for (j=0;j<i;j++)
{
    p1 = 0;
    p2 = 12;
    strcpy((char*)buffer, "a1b2c3d4e5f6");

    retcode = Rockey(RY_WRITE, &handle[j], &lp1, &lp2, &p1, &p2, &p3, &p4, buffer);
    if (retcode)
    {
        ShowERR(retcode);
        return;
    }
    printf("Write:a1b2c3d4e5f6\n");

    p1 = 0;
    p2 = 2;
    memset(buffer, 0, 64);
    retcode = Rockey(RY_READ, &handle[j], &lp1, &lp2, &p1, &p2, &p3, &p4, buffer);
    if (retcode)
    {
        ShowERR(retcode);
        return;
    }
    printf("Read: %s\n", buffer);

    if (!strcmp(buffer, "a1"))
        printf("Run Module 1\n");
    else
        break;
}
```

```

p1 = 2;
p2 = 2;
memset(buffer, 0, 64);
retcode = Rockey(RY_READ, &handle[j], &lp1, &lp2, &p1, &p2, &p3, &p4, buffer);
if (retcode)
{
    ShowERR(retcode);
    return;

}

printf("Read: %s\n", buffer);

if (!strcmp(buffer,"b2"))
    printf("Run Module 2\n");
else
    break;

retcode = Rockey(RY_CLOSE, &handle[j], &lp1, &lp2, &p1, &p2, &p3, &p4, buffer);
if (retcode)
{
    ShowERR(retcode);
    return;
}

printf("\n");
getch();

}

}

```

Step 16: Write a number to the UDZ and decrease it during run time as a means of controlling a software module. We recommended that you use the encryption idea in Step 12 combined with Step 16.

```

#include <windows.h>
#include <stdio.h>
#include <conio.h>
#include "ryvc32.h"

void ShowERR(WORD retcode)
{
    if (retcode == 0) return;
    printf("Error Code: %d\n", retcode);
}

```

```

void main()
{
    WORD handle[16], p1, p2, p3, p4, retcode;

    DWORD lp1, lp2;
    BYTE buffer[1024];

    int i, j,num;

    p1 = 0xc44c;
    p2 = 0xc8f8;
    p3 = 0;
    p4 = 0;

    {

        retcode = Rockey(RY_FIND, &handle[0], &lp1, &lp2, &p1, &p2, &p3, &p4, buffer);
        if (retcode)
        {
            ShowERR(retcode);
            return;
        }
        printf("Find Rock: %08X\n", lp1);

        retcode = Rockey(RY_OPEN, &handle[0], &lp1, &lp2, &p1, &p2, &p3, &p4, buffer);
        if (retcode)
        {
            ShowERR(retcode);
            return;
        }

        i = 1;
        while (retcode == 0)
        {
            retcode = Rockey(RY_FIND_NEXT, &handle[i], &lp1, &lp2, &p1, &p2, &p3, &p4, buffer);
            if (retcode == ERR_NOMORE) break;
            if (retcode)
            {
                ShowERR(retcode);
                return;
            }

            retcode = Rockey(RY_OPEN, &handle[i], &lp1, &lp2, &p1, &p2, &p3, &p4, buffer);
            if (retcode)

```

```

    {
        ShowERR(retcode);
        return;
    }

    i++;
    printf("Find Rock: %08X\n", lp1);
}
printf("\n");

for (j=0;j<i;j++)
{
    p1 = 0;
    p2 = 1;
    strcpy((char*)buffer, "3");
    retcode = Rockey(RY_WRITE, &handle[j], &lp1, &lp2, &p1, &p2, &p3, &p4, buffer);
    if (retcode)
    {
        ShowERR(retcode);
        return;
    }
    printf("Write: 3\n");
    p1 = 0;
    p2 = 1;
    memset(buffer, 0, 64);
    retcode = Rockey(RY_READ, &handle[j], &lp1, &lp2, &p1, &p2, &p3, &p4, buffer);
    if (retcode)
    {
        ShowERR(retcode);
        return;
    }
    printf("Read: %s\n", buffer);

    num=atoi(buffer);

    if(num)
    {
        printf("Hello FeiTian!\n");
        num--;
    }
    else
    {
        return;
    }
    p1 = 0;
    p2 = 1;
    sprintf(buffer, "%ld", num);
    retcode = Rockey(RY_WRITE, &handle[j], &lp1, &lp2, &p1, &p2, &p3, &p4, buffer);
}

```

```
if (retcode)
{
    ShowERR(retcode);
    return;
}
printf("Write: %ld\n",num);

retcode = Rockey(RY_CLOSE, &handle[j], &lp1, &lp2, &p1, &p2, &p3, &p4, buffer);
if (retcode)
{
    ShowERR(retcode);
    return;
}
printf("\n");

}

}
```

2. Seed code advanced applications

Step 17: You may use different seed codes for different software modules or in different places in the application. Then verify the seed codes in the applications.

```
#include <windows.h>
#include <stdio.h>
#include <conio.h>
#include "ryvc32.h"

void ShowERR(WORD retcode)
{
    if (retcode == 0) return;
    printf("Error Code: %d\n", retcode);
}

void main()
{
    WORD handle[16], p1, p2, p3, p4, retcode;
    DWORD lp1, lp2;
    BYTE buffer[1024];

    int i, j;
```

```
p1 = 0xc44c;
p2 = 0xc8f8;
p3 = 0;
p4 = 0;

retcode = Rockey(RY_FIND, &handle[0], &lp1, &lp2, &p1, &p2, &p3, &p4, buffer);
if (retcode)
{
    ShowERR(retcode);
    return;
}
printf("Find Rock: %08X\n", lp1);

retcode = Rockey(RY_OPEN, &handle[0], &lp1, &lp2, &p1, &p2, &p3, &p4, buffer);
if (retcode)
{
    ShowERR(retcode);
    return;
}

i = 1;
while (retcode == 0)
{
    retcode = Rockey(RY_FIND_NEXT, &handle[i], &lp1, &lp2, &p1, &p2, &p3, &p4, buffer);
    if (retcode == ERR_NOMORE) break;
    if (retcode)
    {
        ShowERR(retcode);
        return;
    }

    retcode = Rockey(RY_OPEN, &handle[i], &lp1, &lp2, &p1, &p2, &p3, &p4, buffer);

    if (retcode)
    {
        ShowERR(retcode);
        return;
    }

    i++;
    printf("Find Rock: %08X\n", lp1);
}
printf("\n");
```

```

for (j=0;j<i;j++)
{
    lp2 = 0x12345678;
    retcode = Rockey(RY_SEED, &handle[j], &lp1, &lp2, &p1, &p2, &p3, &p4, buffer);
    if (retcode)
    {
        ShowERR(retcode);
        return;
    }
    printf("Seed: %04X %04X %04X %04X\n", p1, p2, p3, p4);

    if(p1==0xD03A && p2==0x94D6 && p3==0x96A9 && p4==0x7F54)
        printf("Hello Fei!\n");
    else
        break;

    lp2 = 0x87654321;
    retcode = Rockey(RY_SEED, &handle[j], &lp1, &lp2, &p1, &p2, &p3, &p4, buffer);
    if (retcode)
    {
        ShowERR(retcode);
        return;
    }
    printf("Seed: %04X %04X %04X %04X\n", p1, p2, p3, p4);

    if(p1==0xB584 && p2==0xD64F && p3==0xC885 && p4==0x5BA0)
        printf("Hello Tian!\n");
    else
        break;

    lp2 = 0x18273645;
    retcode = Rockey(RY_SEED, &handle[j], &lp1, &lp2, &p1, &p2, &p3, &p4, buffer);
    if (retcode)
    {
        ShowERR(retcode);
        return;
    }
    printf("Seed: %04X %04X %04X %04X\n", p1, p2, p3, p4);

    if(p1==0x2F6D && p2==0x27F8 && p3==0xB3EE && p4==0xBE5A)
        printf("Hello OK!\n");
    else
        break;

    retcode = Rockey(RY_CLOSE, &handle[j], &lp1, &lp2, &p1, &p2, &p3, &p4, buffer);
    if (retcode)
    {
        ShowERR(retcode);
    }
}

```

```

        return;
    }
    printf("\n");
    getch();
}

}

```

In Step 18 we use four outputs of the seed code function to encrypt and decrypt a character string. Be sure that you only include the “decrypt” portion of the code in the application version that is sent to end users.

```

#include <windows.h>
#include <stdio.h>
#include <conio.h>
#include "ryvc32.h"

void ShowERR(WORD retcode)
{
    if (retcode == 0) return;
    printf("Error Code: %d\n", retcode);
}

void main()
{
    char str[20] = "Hello FeiTian!";
    DWORD mykey = 12345678;
    int n, slen;

    WORD handle[16], p1, p2, p3, p4, retcode;
    DWORD lp1, lp2;
    BYTE buffer[1024];

    int i,j;

    p1 = 0xc44c;
    p2 = 0xc8f8;
    p3 = 0x0799;
    p4 = 0xc43b;

    retcode = Rockey(RY_FIND, &handle[0], &lp1, &lp2, &p1, &p2, &p3, &p4, buffer);
    if (retcode)
    {
        ShowERR(retcode);
        return;
    }
}

```

```

    }

    printf("Find Rock: %08X\n", lp1);
    retcode = Rockey(RY_OPEN, &handle[0], &lp1, &lp2, &p1, &p2, &p3, &p4, buffer);
    if (retcode)
    {
        ShowERR(retcode);
        return;
    }

    i = 1;
    while (retcode == 0)
    {
        retcode = Rockey(RY_FIND_NEXT, &handle[i], &lp1, &lp2, &p1, &p2, &p3, &p4, buffer);
        if (retcode == ERR_NOMORE) break;
        if (retcode)
        {
            ShowERR(retcode);
            return;
        }

        retcode = Rockey(RY_OPEN, &handle[i], &lp1, &lp2, &p1, &p2, &p3, &p4, buffer);
        if (retcode)
        {
            ShowERR(retcode);
            return;
        }

        i++;
        printf("Find Rock: %08X\n", lp1);
    }
    printf("\n");

    for (j=0;j<i;j++)
    {
        // Encrypt my data
        slen = strlen(str);
        lp2 = mykey;
        retcode = Rockey(RY_SEED, &handle[j], &lp1, &lp2, &p1, &p2, &p3, &p4, buffer);
        if (retcode) // Error
        {
            printf("Error Code: %d\n", retcode);
            return;
        }

        for (n=0;n<slen;n++)
        {
            str[n] = str[n] + (char)p1 + (char)p2 + (char)p3 + (char)p4;
        }
    }
}

```



```

printf("Encrypted data is %s\n", str);

// Decrypt my data
lp2 = mykey;
retcode = Rockey(RY_SEED, &handle[j], &lp1, &lp2, &p1, &p2, &p3, &p4, buffer);
if (retcode) // Error
{
    printf("Error Code: %d\n", retcode);
    return;
}

for (n=0;n<slen;n++)
{
    str[n] = str[n] - (char)p1 - (char)p2 - (char)p3 - (char)p4;
}
printf("Decrypted data is %s\n", str);

retcode = Rockey(RY_CLOSE, &handle[j], &lp1, &lp2, &p1, &p2, &p3, &p4, buffer);
if (retcode)
{
    ShowERR(retcode);
    return;
}

printf("\n");
getch();
}

}

```

3. User ID advanced applications

Step 19: Some developers will write the current date to the UID when initializing the dongles. During runtime the software may compare the current system time with the date stored in the UID. The program would take appropriate action or continue based on the results of the comparison.

```

#include <windows.h>
#include <stdio.h>
#include <conio.h>
#include "ryvc32.h"

void ShowERR(WORD retcode)
{
    if (retcode == 0) return;
    printf("Error Code: %d\n", retcode);
}

```

```
void main()
{
    WORD handle[16], p1, p2, p3, p4, retcode;
    DWORD lp1, lp2;
    BYTE buffer[1024];
    BYTE buf[1024];

    int i, j;

    SYSTEMTIME st;

    p1 = 0xc44c;
    p2 = 0xc8f8;
    p3 = 0x0799;
    p4 = 0xc43b;

    retcode = Rockey(RY_FIND, &handle[0], &lp1, &lp2, &p1, &p2, &p3, &p4, buffer);
    if (retcode)
    {
        ShowERR(retcode);
        return;
    }
    printf("Find Rock: %08X\n", lp1);
    retcode = Rockey(RY_OPEN, &handle[0], &lp1, &lp2, &p1, &p2, &p3, &p4, buffer);

    if (retcode)
    {
        ShowERR(retcode);
        return;
    }

    i = 1;
    while (retcode == 0)
    {
        retcode = Rockey(RY_FIND_NEXT, &handle[i], &lp1, &lp2, &p1, &p2, &p3, &p4, buffer);
        if (retcode == ERR_NOMORE) break;
        if (retcode)
        {
            ShowERR(retcode);
            return;
        }

        retcode = Rockey(RY_OPEN, &handle[i], &lp1, &lp2, &p1, &p2, &p3, &p4, buffer);
        if (retcode)
        {
            ShowERR(retcode);
        }
    }
}
```

```
        return;
    }

    i++;
    printf("Find Rock: %08X\n", lp1);
}
printf("\n");

for (j=0;j<i;j++)
{
    lp1 = 0x20021101;
    retcode = Rockey(RY_WRITE_USERID, &handle[j], &lp1, &lp2, &p1, &p2, &p3, &p4,
buffer);
    if (retcode)
    {
        ShowERR(retcode);
        return;
    }
    printf("Write User ID: %08X\n", lp1);

    lp1 = 0;
    retcode = Rockey(RY_READ_USERID, &handle[j], &lp1, &lp2, &p1, &p2, &p3, &p4,
buffer);
    if (retcode)
    {
        ShowERR(retcode);
        return;
    }
    printf("Read User ID: %08X\n", lp1);

    sprintf(buffer,"%08X",lp1);
    GetLocalTime(&st);
    printf("Date:%04d%02d%02d\n",st.wYear,st.wMonth,st.wDay);

    sprintf(buf,"%04d%02d%02d",st.wYear,st.wMonth,st.wDay);
    if(strcmp(buf,buffer)>=0)
    {
        printf("ok!\n");
    }
    else

        break;
}
```

```

retcode = Rockey(RY_CLOSE, &handle[j], &lp1, &lp2, &p1, &p2, &p3, &p4, buffer);
if (retcode)
{
    ShowERR(retcode);
    return;
}

printf("\n");
getch();

}
}

```

4. Module advanced applications

Step 20: Module encryption allows you to selectively control portions of your application with the ROCKEY4 modules.

```

#include <windows.h>
#include <stdio.h>
#include <conio.h>
#include "ryvc32.h"

void ShowERR(WORD retcode)
{
    if (retcode == 0) return;
    printf("Error Code: %d\n", retcode);
}

void main()
{
    WORD handle[16], p1, p2, p3, p4, retcode;
    DWORD lp1, lp2;
    BYTE buffer[1024];

    int i, j;

    p1 = 0xc44c;
    p2 = 0xc8f8;
    p3 = 0x0799;
    p4 = 0xc43b;

    {
        retcode = Rockey(RY_FIND, &handle[0], &lp1, &lp2, &p1, &p2, &p3, &p4, buffer);

```

```
if (retcode)
{
    ShowERR(retcode);
    return;
}
printf("Find Rock: %08X\n", lp1);

retcode = Rockey(RY_OPEN, &handle[0], &lp1, &lp2, &p1, &p2, &p3, &p4, buffer);
if (retcode)
{
    ShowERR(retcode);
    return;
}

i = 1;
while (retcode == 0)
{
    retcode = Rockey(RY_FIND_NEXT, &handle[i], &lp1, &lp2, &p1, &p2, &p3, &p4, buffer);
    if (retcode == ERR_NOMORE) break;
    if (retcode)
    {
        ShowERR(retcode);
        return;
    }

    retcode = Rockey(RY_OPEN, &handle[i], &lp1, &lp2, &p1, &p2, &p3, &p4, buffer);
    if (retcode)
    {
        ShowERR(retcode);
        return;
    }

    i++;
    printf("Find Rock: %08X\n", lp1);
}
printf("\n");

for (j=0;j<i;j++)
{
    p1 = 0;
    p2 = 0x2121;
    p3 = 0;
    retcode = Rockey(RY_SET_MOUDLE, &handle[j], &lp1, &lp2, &p1, &p2, &p3, &p4, buffer);
    if (retcode)
    {
        ShowERR(retcode);
        return;
    }
}
```

```
printf("Set Moudle 0: Pass = %04X Decrease no allow\n", p2);

p1 = 0;
retcode = Rockey(RY_CHECK_MOUDLE, &handle[j], &lp1, &lp2, &p1, &p2, &p3, &p4, buffer);
if (retcode)
{
    ShowERR(retcode);
    return;
}
printf("Check Moudle 0: ");
if (p2)
    printf("Run Modul 1!\n");
else
    break;

printf("\n");

p1 = 8;
p2 = 0xFFFF;
p3 = 0;
retcode = Rockey(RY_SET_MOUDLE, &handle[j], &lp1, &lp2, &p1, &p2, &p3, &p4, buffer);
if (retcode)
{
    ShowERR(retcode);
    return;
}
printf("Set Moudle 8: Pass = %04X Decrease no allow\n", p2);

p1 = 8;
retcode = Rockey(RY_CHECK_MOUDLE, &handle[j], &lp1, &lp2, &p1, &p2, &p3, &p4, buffer);
if (retcode)
{
    ShowERR(retcode);
    return;
}
printf("Check Moudle 8: ");
if (p2)
    printf("Run Modul 2!");
else
    break;

retcode = Rockey(RY_CLOSE, &handle[j], &lp1, &lp2, &p1, &p2, &p3, &p4, buffer);
if (retcode)
{
    ShowERR(retcode);
```

```
        return;  
    }  
  
    printf("\n");  
  
}  
}  
}
```

Step 21: This program discussed how to perform multi-module encryption and check the status of the modules. Many applications are segmented into program modules that users may choose or purchase separately. For example , a user may purchase three of four available application modules and the licensing policy would allow the user to execute only those modules that were purchased. ROCKEY4 modules may be used to enforce this licensing scheme.

But what if the user later decides to purchase the fourth module? Some manufacturers require that the original dongle be returned for replacement. This is obviously not convenient from the user's perspective. ROCKEY4 dongles, on the other hand, may be cascaded so that access to the fourth application module may be added by adding another dongle to the LPT port.

```
#include <windows.h>  
#include <stdio.h>  
#include "ryvc32.h"           // Include Rockey Header File  
  
void main()  
{  
int i, j, rynum;  
WORD retcode;  
DWORD HID[16];  
  
WORD handle[16], p1, p2, p3, p4;    // Rockey Variable  
DWORD lp1, lp2;                    // Rockey Variable  
BYTE buffer[1024];                 // Rockey Variable  
  
p1 = 0xc44c;          // Rockey Demo Password 1  
p2 = 0xc8f8;          // Rockey Demo Password 2  
p3 = 0;               // Program needn't Password 3, Set to 0  
p4 = 0;               // Program needn't Password 4, Set to 0  
  
// Try to find all Rockey  
for (i=0;i<16;i++)  
{  
    if (0 == i)  
    {  
        retcode = Rockey(RY_FIND, &handle[i], &lp1, &lp2, &p1, &p2, &p3, &p4, buffer);  
    }  
}
```

```

        if (retcode == ERR_NOMORE) break;
    }
else
{
    // Notice : lp1 = Last found hardID
    retcode = Rockey(RY_FIND_NEXT, &handle[i], &lp1, &lp2, &p1, &p2, &p3, &p4, buffer);
    if (retcode == ERR_NOMORE) break;
}

if (retcode) // Error
{
    printf("Error Code: %d\n", retcode);
    return;
}

printf("Found Rockey: %08X\n", lp1);
HID[i] = lp1; // Save HardID
retcode = Rockey(RY_OPEN, &handle[i], &lp1, &lp2, &p1, &p2, &p3, &p4, buffer);
if (retcode) // Error
{
    printf("Error Code: %d\n", retcode);

    return;
}
printf("\n");

rynum = i;

// Do our work
for (i=0;i<rynum;i++)
{
    printf("Rockey %08X module status: ", HID[i]);
    for (j=0;j<16;j++)
    {
        p1 = j; // Module No
        retcode = Rockey(RY_CHECK_MOUDLE, &handle[i], &lp1, &lp2, &p1, &p2, &p3, &p4,
buffer);
        if (retcode) // Error
        {
            printf("Error Code: %d\n", retcode);
            return;
        }
        if (p2) printf("O");
        else printf("X");
    }
    printf("\n");
}

```

```
// Close all opened Rockey
for (i=0;i<rynum;i++)
{
    retcode = Rockey(RY_CLOSE, &handle[i], &lp1, &lp2, &p1, &p2, &p3, &p4, buffer);
    if (retcode)
    {
        printf("Error Code: %d\n", retcode);
        return;
    }
}
```

The above program searches all dongles with the same passwords attached to the computer and displays the status of every module in every listed dongle. “ O” means that the module may be used and is not zero; “ X” means that the module cannot be used. In a protection scheme that relies on ROCKEY4 modules this program would help the developer identify modules that are usable from ones that should be terminated.

3. The same code dongle advanced applications

If you have several software products but only a single purchase code – meaning that the passwords are all the same – you may use the solution indicated below to differentiate the dongles.

In Step 22 the UDZ is used to distinguish the dongles with the same passwords. For example, the dongles with UDZ content of “ Ver 10” correspond to software A.

```
#include <windows.h>
#include <stdio.h>
#include <conio.h>
#include "ryvc32.h"

void ShowERR(WORD retcode)
{
    if (retcode == 0) return;
    printf("Error Code: %d\n", retcode);
}

void main()
{
    WORD handle[16], p1, p2, p3, p4, retcode;
    WORD handleEnd;
    DWORD lp1, lp2;
    BYTE buffer[1024];

    int i, j;

    p1 = 0xc44c;
```

```
p2 = 0xc8f8;
p3 = 0x0799;
p4 = 0xc43b;

{

retcode = Rockey(RY_FIND, &handle[0], &lp1, &lp2, &p1, &p2, &p3, &p4, buffer);

if (retcode)
{
    ShowERR(retcode);
    return;
}
printf("Find Rock: %08X\n", lp1);

retcode = Rockey(RY_OPEN, &handle[0], &lp1, &lp2, &p1, &p2, &p3, &p4, buffer);
if (retcode)
{
    ShowERR(retcode);
    return;
}

i = 1;
while (retcode == 0)
{
    retcode = Rockey(RY_FIND_NEXT, &handle[i], &lp1, &lp2, &p1, &p2, &p3, &p4, buffer);
    if (retcode == ERR_NOMORE) break;
    if (retcode)
    {
        ShowERR(retcode);
        return;
    }

    retcode = Rockey(RY_OPEN, &handle[i], &lp1, &lp2, &p1, &p2, &p3, &p4, buffer);
    if (retcode)
    {
        ShowERR(retcode);
        return;
    }

    i++;
    printf("Find Rock: %08X\n", lp1);
}
printf("\n");

for (j=0;j<i;j++)
{
```



```

/*p1 = 0;

p2 = 5;
strcpy((char*)buffer, "Ver10");
retcode = Rockey(RY_WRITE, &handle[j], &lp1, &lp2, &p1, &p2, &p3, &p4, buffer);
if (retcode)
{
    ShowERR(retcode);
    return;
}
printf("Write:%s\n",buffer);
*/

```



```

p1 = 0;
p2 = 5;
memset(buffer, 0, 64);
retcode = Rockey(RY_READ, &handle[j], &lp1, &lp2, &p1, &p2, &p3, &p4, buffer);
if (retcode)
{
    ShowERR(retcode);
    return;
}
printf("Read: %s\n", buffer);

if (!strcmp(buffer,"Ver10"))
{
    handleEnd=handle[j];
    break;
}

```



```

}

{
//=====A=====
retcode = Rockey(RY_RANDOM, &handleEnd, &lp1, &lp2, &p1, &p2, &p3, &p4, buffer);
if (retcode)
{
    ShowERR(retcode);
    return;
}
printf("Random: %04X\n", p1);

lp2 = 0x12345678;
retcode = Rockey(RY_SEED, &handleEnd, &lp1, &lp2, &p1, &p2, &p3, &p4, buffer);
if (retcode)
{

```

```

        ShowERR(retcode);
        return;
    }
    printf("Seed: %04X %04X %04X %04X\n", p1, p2, p3, p4);

    retcode = Rockey(RY_CLOSE, &handleEnd, &lp1, &lp2, &p1, &p2, &p3, &p4, buffer);
    if (retcode)
    {
        ShowERR(retcode);
        return;
    }

    printf("\n");
}

}
}
}

```

In Step 23 the UID is used to distinguish the dongles with the same passwords. For example, dongles with UID of “ 11111111” (hexadecimal) correspond to software A.

```

#include <windows.h>
#include <stdio.h>
#include <conio.h>
#include "ryvc32.h"

void ShowERR(WORD retcode)
{
    if (retcode == 0) return;
    printf("Error Code: %d\n", retcode);
}

void main()
{
    WORD handle[16], p1, p2, p3, p4, retcode;
    WORD handleEnd;
    DWORD lp1, lp2;
    BYTE buffer[1024];

    int i, j;

    p1 = 0xc44c;
    p2 = 0xc8f8;
    p3 = 0x0799;
    p4 = 0xc43b;
}

```

```

{
    retcode = Rockey(RY_FIND, &handle[0], &lp1, &lp2, &p1, &p2, &p3, &p4, buffer);
    if (retcode)
    {
        ShowERR(retcode);
        return;
    }
    printf("Find Rock: %08X\n", lp1);

    retcode = Rockey(RY_OPEN, &handle[0], &lp1, &lp2, &p1, &p2, &p3, &p4, buffer);
    if (retcode)
    {
        ShowERR(retcode);
        return;
    }

    i = 1;
    while (retcode == 0)
    {
        retcode = Rockey(RY_FIND_NEXT, &handle[i], &lp1, &lp2, &p1, &p2, &p3, &p4, buffer);
        if (retcode == ERR_NOMORE) break;
        if (retcode)
        {
            ShowERR(retcode);

            return;
        }

        retcode = Rockey(RY_OPEN, &handle[i], &lp1, &lp2, &p1, &p2, &p3, &p4, buffer);
        if (retcode)
        {
            ShowERR(retcode);
            return;
        }

        i++;
        printf("Find Rock: %08X\n", lp1);
    }
    printf("\n");

    for (j=0;j<i;j++)
    {
        /*lp1= 0x11111111;
        retcode = Rockey(RY_WRITE_USERID, &handle[j], &lp1, &lp2, &p1, &p2, &p3, &p4, buffer);
        if (retcode)
        {

```

```
        ShowERR(retcode);
        return;
    }
    printf("Write User ID: %08X\n", lp1);
}

lp1 = 0;
retcode = Rockey(RY_READ_USERID, &handle[j], &lp1, &lp2, &p1, &p2, &p3, &p4, buffer);
if (retcode)
{
    ShowERR(retcode);
    return;
}
if(lp1==0x11111111)
{
    handleEnd=handle[j];
    break;
}
}

{

//=====A=====
p1 = 0;
p2 = 12;
strcpy((char*)buffer, "Hello Feitian!");
retcode = Rockey(RY_WRITE, &handleEnd, &lp1, &lp2, &p1, &p2, &p3, &p4, buffer);
if (retcode)
{
    ShowERR(retcode);
    return;
}
printf("Write: %s\n",buffer);

p1 = 0;
p2 = 12;
buffer[512]=0;
retcode = Rockey(RY_READ, &handleEnd, &lp1, &lp2, &p1, &p2, &p3, &p4, buffer);
if (retcode)
{
    ShowERR(retcode);
    return;
}
printf("Read: %s\n",buffer);

retcode = Rockey(RY_RANDOM, &handleEnd, &lp1, &lp2, &p1, &p2, &p3, &p4, buffer);
if (retcode)
{
    ShowERR(retcode);
```

```
        return;
    }
    printf("Random: %04X\n", p1);

    lp2 = 0x12345678;
    retcode = Rockey(RY_SEED, &handleEnd, &lp1, &lp2, &p1, &p2, &p3, &p4, buffer);
    if (retcode)
    {
        ShowERR(retcode);
        return;
    }
    printf("Seed: %04X %04X %04X %04X\n", p1, p2, p3, p4);

    retcode = Rockey(RY_CLOSE, &handleEnd, &lp1, &lp2, &p1, &p2, &p3, &p4, buffer);
    if (retcode)
    {
        ShowERR(retcode);
        return;
    }

    printf("\n");
}

}
```

Chapter 8

ROCKEY4 Hardware Algorithms

Developers may define their own algorithms and securely store them inside ROCKEY4. The dongle may then be used to calculate a result, and the result used by the application. Since the ROCKEY4's User Algorithm Zone (UAZ) is unreadable, even by the manufacturer, this type of software protection is potentially very powerful.

Developers may use either the ROCKEY editor or the RY_WRITE_ARITHMETIC function to write algorithms to the dongle.

8.1 ROCKEY User Defined Algorithm Introduction

8.1.1 Instruction Format

All instructions must be of the form:

`reg1 = reg2 op reg3/value`

reg1, reg2 and reg3 are registers, value is a figure, op is an operator.

For example: `A = A + B`

ROCKEY supports the following operations:

- + Addition
 - Subtraction
 - < Cyclic left shift
 - *
 - Multiplication
 - ^ XOR
 - & And
 - | Or
 - ~ NEG. (ROCKEY4 Standard)
 - ? Compare (ROCKEY4 Plus and NetROCKEY4) 
- value* is a decimal figure between 0 and 63.

Note:

” ~” is just for ROCKEY4 Standard. ROCKEY4 Plus and NetROCKEY4 use “?” for comparison.

1. “ ~” operator is a unary. Therefore `A=~A` is written `A = A ~B`, where the parameter B is ignored.



2. “?” operator is for comparison, for example, $C = A ? B$, the results are listed below:

C	A?B	B?A
A < B	0	FFFF
A = B	FFFF	FFFF
A > B	FFFF	0

It will write either “ 0xFFFF” or “ 0” to parameter C according to the table above.

First let us have a look at the algorithm example we will write to ROCKEY:

$A = A + B$, $B = B + E$, $C = A * F$, $D = B + C$, $H = H ^ H$

A, B, C... are registers in ROCKEY. There are a total of eight 16-bit registers in ROCKEY and they are designed: A, B, C, D, E, F, G and H.

8.1.2 Internal Algorithms & Application Interface

Feitian offers 3 calculation functions to call the user-defined algorithms:

`RY_CALCULATE1`, `RY_CALCULATE2`, `RY_CALCULATE3`

These three functions are structurally similar. Data is passed and received by way of the memory addresses $p1$, $p2$, $p3$, and $p4$.

When passing data to registers:

- Register A = $p1$
- Register B = $p2$
- Register C = $p3$
- Register D = $p4$

Register variables vary according to the calculation type:

- Register E
- Register F
- Register G
- Register H

When receiving data from registers:

- $p1$ = Register A
- $p2$ = Register B
- $p3$ = Register C
- $p4$ = Register D

Register A, B, C and D are user interface variables, register E, F, G and H are internal variables.



8.1.3 Differences between the Three Functions

p1, p2, p3 and p4 correspond to registers A, B, C and D in all three calculation functions. These registers are used nearly identically by the three calculation functions. The differences between the functions can be seen by reviewing the results written to registers E, F, G and H.

When a developer's ROCKEY4 internal program is called, registers A, B, C and D will be populated with data from p1, p2, p3 and p4. The content of registers E, F, G and H will be initialized according to the calculation function in use. See below:

Variable	RY_CALCULATE1
A	P1
B	P2
C	P3
D	P4
E	HiWord of hardware ID
F	LoWord of hardware ID
G	Value stored in module *lp2
H	Random number

Variable	RY_CALCULATE2
A	P1
B	P2
C	P3
D	P4
E	Seed Result 1
F	Seed Result 2
G	Seed Result 3
H	Seed Result 4

Variable	RY_CALCULATE3
A	P1
B	P2
C	P3
D	P4
E	Value in module *lp2
F	Value in module (*lp2 + 1)
G	Value in module (*lp2 + 2)
H	Value in module (*lp2 + 3)



8.1.4 API Interface of the User's Applications

Below is the definition and description of the three calculation functions.

Function	RY_CALCULATE1 (Calculation 1)
Objective	Perform specified calculation
Input parameters	function = RY_CALCULATE1 *handle = ROCKEY's handle *lp1 = Start point of calculation *lp2 = Module number *p1 = Input value 1 *p2 = Input value 2 *p3 = Input value 3 *p4 = Input value 4
Return value	A return value = “ 0” indicates that the function worked correctly. Any other return value indicates an error. When success, *p1 = Return value 1 *p2 = Return value 2 *p3 = Return value 3 *p4 = Return value 4
Note	If the internal algorithm is $A = B + C$, then the call result is $*p1 = *p2 + *p3$. For example: The internal algorithm is $A = A + G$, if $*p1 = 0$, then when returning you may guess the content of module $*p1 = *lp2$. Though you cannot read the content of modules directly, you may determine the content by algorithm. If possible, you had better check the content with an algorithm, not only compare in the program.

Function	RY_CALCULATE2 (Calculation 2)
Objective	Perform specified calculation
Input parameter	function = RY_CALCULATE2 *handle = ROCKEY4's handle *lp1 = Start point of calculation *lp2 = Seed Code *p1 = Input value 1 *p2 = Input value 2 *p3 = Input value 3 *p4 = Input value 4
Return value	A return value = “ 0” indicates that the function worked correctly. Any other return value indicates an error. When success,

	*p1 = Return value 1 *p2 = Return value 2 *p3 = Return value 3 *p4 = Return value 4
Note	When performing calculation 2, the initial values of register E, F, G and H are the return values of seed code *lp2, to make it simple, ROCKEY calls function RY_SEED with seed code *lp2, and writes the return values to register E, F, G and H for next operation.

Function	RY_CALCULATE3 (Calculation 3)
Objective	Perform specified calculation
Input parameter	function = RY_CALCULATE3 *handle = ROCKEY4's handle *lp1 = Start point of calculation *lp2 = Module number *p1 = Input value 1 *p2 = Input value 2 *p3 = Input value 3 *p4 = Input value 4
Return value	A return value = “ 0” indicates that the function worked correctly. Any other return value indicates an error. When success, *p1 = Return value 1 *p2 = Return value 2 *p3 = Return value 3 *p4 = Return value 4
Note	When performing calculation 3, the initial values of register E, F, G and H are the content of module *lp2 and *lp2+1/2/3, for example: When calls calculation 3 with *lp2 = 0, the initial values of register E, F, G and H are: E = Content of module 0 F = Content of module 1 G = Content of module 2 H = Content of module 3 <i>Note: The address will return to “ 0” when the module address call exceeds 15. For example: Calculation 3 with *lp2 = 14, the initial values of register E, F, G and H are:</i> <i>E = Content of module 14</i> <i>F = Content of module 15</i> <i>G = Content of module 0</i> <i>H = Content of module 1</i>

8.2 Writing User Defined Algorithms into ROCKEY

8.2.1 Writing Algorithm

Developers may use the *RY_WRITE_ARITHMETIC* algorithm to write algorithms to the ROCKEY4 User Algorithm Zone (UAZ). The ROCKEY editor is another option for writing algorithms to the UAZ.

Function	<i>RY_WRITE_ARITHMETIC</i> (Write algorithm)
Objective	Write user defined algorithm to ROCKEY
Input parameter	function = <i>RY_WRITE_ARITHMETIC</i> *handle = ROCKEY4's handle *p1 = Start point of calculation *buffer = Instruction string
Return	A return value = “ 0” indicates that the function worked correctly. Any other return value indicates an error.

For example:

```
strcpy(buffer, "A=A+E, A=A+F, A=A+G, A=A+H");
p1 = 3;
retcode = Rockey(RY_WRITE_ARITHMETIC, handle, &p1, &p2, &p1, &p2, &p3, &p4,
buffer);
```

The “buffer” is the place for you to temporarily store the algorithm One instruction is separated from another by a “,”. ROCKEY will automatically assign the first instruction in the algorithm, “Start” and the last instruction, “End”. Taking this program as an example:

```
Address 3 in Algorithm Zone: A=A+E
Address 4 in Algorithm Zone: A=A+F
Address 5 in Algorithm Zone: A=A+G
Address 6 in Algorithm Zone: A=A+H
```

Then 3 is the starting point of the algorithm in the User Algorithm Zone (UAZ). 6 is the end point. ROCKEY will return to the user application after performing the instruction in address 6. The users must call the program in the dongles from the starting point of the algorithm Otherwise the results are 4 random numbers.

8.2.2 Instruction Conventions

There are some conventions when developers write algorithm instructions:



A = A + B	Valid instruction
D = D ^ D	Valid instruction
A = B	Invalid instruction, A = B B would be correct.
A = 0	Invalid instruction, A = A ^ A would be correct
C = 3 * B	Invalid instruction, C = B * 3 would be correct
D = 3 + 4	Invalid instruction, there can not be two constants
A = A / B	Invalid instruction, ROCKEY does not support division operator
H = E*200	Invalid instruction, constant must be less than 64
A = A*63	If it is the first or last instruction it is an invalid instruction, otherwise valid.

8.3 User Defined Algorithm Examples

8.3.1 Basic Algorithm Application Examples

Calculation 1 example

First we write the algorithm (We only need to write the algorithm once. The code used to write the algorithm(s) to the dongle does not appear in the application delivered to the end user.)

```
p1 = 0;  
strcpy(buffer, "H=H^H, A=A*23, F=B*17, A=A+F, A=A+G, A=A<C, A=A^D, B=B^B, C=C^C,  
D=D^D");  
retcode = Rockey(RY_WRITE_ARITHMETIC, handle, &lp1, &lp2, &p1, &p2, &p3, &p4, buffer);
```

Then call this algorithm from the program:

```
lp1 = 0;           // Start point of calculation  
lp2 = 7;          // Module number  
p1 = 5;           // Initial value of A  
p2 = 3;           // Initial value of B  
p3 = 1;           // Initial value of C  
p4 = 0xffff;      // Initial value of D  
retcode = Rockey(RY_CALCULATE1, handle, &lp1, &lp2, &p1, &p2, &p3, &p4, buffer);
```

The command begins to execute from instruction 0 (lp1) of the UAZ and the registers are initialized as follows:

A = 5 (p1)
B = 3 (p2)
C = 1 (p3)
D = 0xffff (p4)
E = the upper 16-bit of HID
F = the lower 16-bit of HID
G = the value in module #7 (lp2)
H = random number (16-bit)



Assuming that the value in module 7 is 0x2121, the result of this calculation will be:
 $((5*23 + 3*17 + 0x2121) < 1) \wedge 0xffff = 0xbc71$

Calculation 1 example codes – Step 24:

```
#include <windows.h>
#include <stdio.h>
#include <conio.h>
#include "ryvc32.h"

void ShowERR(WORD retcode)
{
    if (retcode == 0) return;
    printf("Error Code: %d\n", retcode);
}

void main()
{
    WORD handle[16], p1, p2, p3, p4, retcode;
    DWORD lp1, lp2;
    BYTE buffer[1024];

    int i, j;

    char cmd[] = "H=H^H, A=A*23, F=B*17, A=A+F, A=A+G, A=A<C, A=A^D, B=B^B, C=C^C,
D=D^D";
    p1 = 0xc44c;
    p2 = 0xc8f8;
    p3 = 0x0799;
    p4 = 0xc43b;

    retcode = Rockey(RY_FIND, &handle[0], &lp1, &lp2, &p1, &p2, &p3, &p4, buffer);
    if (retcode)
    {
        ShowERR(retcode);
        return;
    }
    printf("Find Rock: %08X\n", lp1);

    retcode = Rockey(RY_OPEN, &handle[0], &lp1, &lp2, &p1, &p2, &p3, &p4, buffer);
    if (retcode)
    {
        ShowERR(retcode);
        return;
    }
```

ROCKEY

```
i = 1;
while (retcode == 0)
{
    retcode = Rockey(RY_FIND_NEXT, &handle[i], &lp1, &lp2, &p1, &p2, &p3, &p4, buffer);
    if (retcode == ERR_NOMORE) break;
    if (retcode)
    {
        ShowERR(retcode);
        return;
    }

    retcode = Rockey(RY_OPEN, &handle[i], &lp1, &lp2, &p1, &p2, &p3, &p4, buffer);
    if (retcode)

    {
        ShowERR(retcode);
        return;
    }

    i++;

    printf("Find Rock: %08X\n", lp1);
}
printf("\n");

for (j=0;j<i;j++)
{
/*
p1 = 7;
p2 = 0x2121;
p3 = 0;
retcode = Rockey(RY_SET_MOUDLE, &handle[j], &lp1, &lp2, &p1, &p2, &p3, &p4, buffer);
if (retcode)
{
    ShowERR(retcode);
    return;
}
printf("Set Moudle 7: Pass = %04X Decrease no allow\n", p2);
printf("\n");
*/
p1 = 0;
strcpy((char*)buffer, cmd);
retcode = Rockey(RY_WRITE_ARITHMETIC, &handle[j], &lp1, &lp2, &p1, &p2, &p3, &p4,
buffer);
if (retcode)
{
```

```

        ShowERR(retcode);
        return;
    }
    printf("Write Arithmetic 1\n");

    lp1 = 0;
    lp2 = 7;

    p1 = 5;
    p2 = 3;
    p3 = 1;
    p4 = 0xffff;
    retcode = Rockey(RY_CALCULATE1, &handle[j], &lp1, &lp2, &p1, &p2, &p3, &p4, buffer);
    if (retcode)
    {
        ShowERR(retcode);
        return;
    }
    printf("Calculate Input: p1=5, p2=3, p3=1, p4=0xffff\n");

    printf("\n");
    printf("Result = ((5*23 + 3*17 + 0x2121) < 1) ^ 0xffff = 0xBC71\n");
    printf("Calculate Output: p1=%x, p2=%x, p3=%x, p4=%x\n", p1, p2, p3, p4);

    retcode = Rockey(RY_CLOSE, &handle[j], &lp1, &lp2, &p1, &p2, &p3, &p4, buffer);
    if (retcode)
    {
        ShowERR(retcode);
        return;
    }

    printf("\n");
    getch();
}
}

```

Calculation 2 example

In Step 25 we write algorithm ("A=A+B, A=A+C, A=A+D, A=A+E, A=A+F, A=A+G, A=A+H") to the UAZ, and the calculation result is 0x7b17.

```
#include <windows.h>
#include <stdio.h>
#include <conio.h>
#include "ryvc32.h"

void ShowERR(WORD retcode)
{
    if (retcode == 0) return;
    printf("Error Code: %d\n", retcode);
}

void main()
{
    WORD handle[16], p1, p2, p3, p4, retcode;
    DWORD lp1, lp2;
    BYTE buffer[1024];

    int i, j;

    char cmd1[] = "A=A+B, A=A+C, A=A+D, A=A+E, A=A+F, A=A+G, A=A+H";

    p1 = 0xc44c;
    p2 = 0xc8f8;
    p3 = 0x0799;
    p4 = 0xc43b;

    retcode = Rockey(RY_FIND, &handle[0], &lp1, &lp2, &p1, &p2, &p3, &p4, buffer);
    if (retcode)
    {
        ShowERR(retcode);
        return;
    }
    printf("Find Rock: %08X\n", lp1);

    retcode = Rockey(RY_OPEN, &handle[0], &lp1, &lp2, &p1, &p2, &p3, &p4, buffer);
    if (retcode)
    {
        ShowERR(retcode);
        return;
    }

    i = 1;
    while (retcode == 0)
    {
        retcode = Rockey(RY_FIND_NEXT, &handle[i], &lp1, &lp2, &p1, &p2, &p3, &p4, buffer);
        if (retcode == ERR_NOMORE) break;
    }
}
```

```

if (retcode)
{
    ShowERR(retcode);
    return;
}

retcode = Rockey(RY_OPEN, &handle[i], &lp1, &lp2, &p1, &p2, &p3, &p4, buffer);
if (retcode)
{
    ShowERR(retcode);
    return;
}

i++;

printf("Find Rock: %08X\n", lp1);
}
printf("\n");

for (j=0;j<i;j++)
{
    /*
    lp2 = 0x12345678;
    retcode = Rockey(RY_SEED, &handle[j], &lp1, &lp2, &p1, &p2, &p3, &p4, buffer);
    if (retcode)
    {
        ShowERR(retcode);
        return;
    }
    printf("Seed: %04X %04X %04X %04X\n", p1, p2, p3, p4);
    printf("\n");
    */

    p1 = 10;
    strcpy((char*)buffer, cmd1);
    retcode = Rockey(RY_WRITE_ARITHMETIC, &handle[j], &lp1, &lp2, &p1, &p2, &p3,
&p4, buffer);
    if (retcode)
    {
        ShowERR(retcode);
        return;
    }
    printf("Write Arithmetic 2\n");

    lp1 = 10;
}

```

```
lp2 = 0x12345678;
p1 = 1;
p2 = 2;
p3 = 3;
p4 = 4;
retcode = Rockey(RY_CALCULATE2, &handle[j], &lp1, &lp2, &p1, &p2, &p3, &p4,
buffer);
if (retcode)
{
    ShowERR(retcode);
    return;
}
printf("Calculate Input: p1=1, p2=2, p3=3, p4=4\n");

printf("\n");
printf("Result =d03a + 94d6 + 96a9 + 7f54 + 1 + 2 + 3 + 4=0x7b17\n");
printf("Calculate Output: p1=%x, p2=%x, p3=%x, p4=%x\n", p1, p2, p3, p4);

retcode = Rockey(RY_CLOSE, &handle[j], &lp1, &lp2, &p1, &p2, &p3, &p4, buffer);
if (retcode)
{
    ShowERR(retcode);
    return;
}

printf("\n");
getch();
}
```

Calculation 3 example

In Step 26 we write algorithm ("A=A+B, A=A+C, A=A+D, A=A+E, A=A+F, A=A+G, A=A+H") to UAZ, and the calculation result is 0x14.

```
#include <windows.h>
#include <stdio.h>
#include <conio.h>
#include "ryvc32.h"

void ShowERR(WORD retcode)
{
    if (retcode == 0) return;
    printf("Error Code: %d\n", retcode);
}

void main()
{
```

```
WORD handle[16], p1, p2, p3, p4, retcode;
DWORD lp1, lp2;
BYTE buffer[1024];

int i, j;

char cmd2[] = "A=A+B, A=A+C, A=A+D, A=A+E, A=A+F, A=A+G, A=A+H";

p1 = 0xc44c;
p2 = 0xc8f8;
p3 = 0x0799;
p4 = 0xc43b;

retcode = Rockey(RY_FIND, &handle[0], &lp1, &lp2, &p1, &p2, &p3, &p4, buffer);
if (retcode)
{
    ShowERR(retcode);
    return;
}
printf("Find Rock: %08X\n", lp1);

retcode = Rockey(RY_OPEN, &handle[0], &lp1, &lp2, &p1, &p2, &p3, &p4, buffer);
if (retcode)

{
    ShowERR(retcode);
    return;
}

i = 1;
while (retcode == 0)
{
    retcode = Rockey(RY_FIND_NEXT, &handle[i], &lp1, &lp2, &p1, &p2, &p3, &p4, buffer);
    if (retcode == ERR_NOMORE) break;
    if (retcode)
    {
        ShowERR(retcode);
        return;
    }

    retcode = Rockey(RY_OPEN, &handle[i], &lp1, &lp2, &p1, &p2, &p3, &p4, buffer);
    if (retcode)
    {
        ShowERR(retcode);
        return;
    }
}
```

```
i++;

    printf("Find Rock: %08X\n", lp1);
}
printf("\n");

for (j=0;j<i;j++)
{
/*
p1 = 0;
p2 = 1;
p3 = 0;
retcode = Rockey(RY_SET_MOUDLE, &handle[j], &lp1, &lp2, &p1, &p2, &p3, &p4,
buffer);
if (retcode)
{
    ShowERR(retcode);

    return;
}
printf("Set Moudle 0: Pass = %04X Decrease no allow\n", p2);

p1 = 1;
p2 = 2;
p3 = 0;
retcode = Rockey(RY_SET_MOUDLE, &handle[j], &lp1, &lp2, &p1, &p2, &p3, &p4,
buffer);
if (retcode)
{
    ShowERR(retcode);
    return;
}
printf("Set Moudle 1: Pass = %04X Decrease no allow\n", p2);

p1 = 2;
p2 = 3;
p3 = 0;
retcode = Rockey(RY_SET_MOUDLE, &handle[j], &lp1, &lp2, &p1, &p2, &p3, &p4,
buffer);
if (retcode)
{
    ShowERR(retcode);
    return;
}
printf("Set Moudle 2: Pass = %04X Decrease no allow\n", p2);

p1 = 3;
```

```
p2 = 4;
p3 = 0;
retcode = Rockey(RY_SET_MOUDLE, &handle[j], &lp1, &lp2, &p1, &p2, &p3, &p4,
buffer);
if (retcode)
{
    ShowERR(retcode);
    return;
}
printf("Set Moudle 3: Pass = %04X Decrease no allow\n", p2);
printf("\n");
*/
p1 = 17;
strcpy((char*)buffer, cmd2);
retcode = Rockey(RY_WRITE_ARITHMETIC, &handle[j], &lp1, &lp2, &p1, &p2, &p3,
&p4, buffer);
if (retcode)
{
    ShowERR(retcode);
    return;
}
printf("Write Arithmetic 3\n");

lp1 = 17;
lp2 = 0;
p1 = 1;
p2 = 2;
p3 = 3;
p4 = 4;
retcode = Rockey(RY_CALCULATE3, &handle[j], &lp1, &lp2, &p1, &p2, &p3, &p4,
buffer);
if (retcode)
{
    ShowERR(retcode);
    return;
}
printf("Calculate Input: p1=1, p2=2, p3=3, p4=4\n");

printf("\n");
printf("Result = 1+2+3+4+1+2+3+4=0x14\n");
printf("Calculate Output: p1=%x, p2=%x, p3=%x, p4=%x\n", p1, p2, p3, p4);

retcode = Rockey(RY_CLOSE, &handle[j], &lp1, &lp2, &p1, &p2, &p3, &p4, buffer);
if (retcode)
{
    ShowERR(retcode);
    return;
}
```

```

        printf("\n");
        getch();
    }
}

```

8.3.2 Complex Algorithm Application Examples

Complex example 1

In Step 27 we first search the dongle and get its hardware ID. Then we use the calculation 1 function in the program to get the hardware ID again. Compare the two hardware IDs. If they are different the program will be terminated.

```

#include <windows.h>
#include <stdio.h>
#include <conio.h>
#include "ryvc32.h"

void ShowERR(WORD retcode)
{
    if (retcode == 0) return;
    printf("Error Code: %d\n", retcode);
}

void main()
{
    WORD handle[16], p1, p2, p3, p4, retcode;
    DWORD findlp1,truelp1;

    DWORD lp1, lp2;
    BYTE buffer[1024];

    int i, j;

    char cmd[] = "A=E|E,B=F|F,C=G|G,D=H|H";

    p1 = 0xc44c;
    p2 = 0xc8f8;
    p3 = 0x0799;
    p4 = 0xc43b;

    retcode = Rockey(RY_FIND, &handle[0], &lp1, &lp2, &p1, &p2, &p3, &p4, buffer);
    if (retcode)

```

```

    {
        ShowERR(retcode);
        return;
    }

    printf("Find Rock: %08X\n", lp1);
    findlp1=lp1;

    retcode = Rockey(RY_OPEN, &handle[0], &lp1, &lp2, &p1, &p2, &p3, &p4, buffer);
    if (retcode)
    {
        ShowERR(retcode);
        return;
    }

    i = 1;
    while (retcode == 0)
    {
        retcode = Rockey(RY_FIND_NEXT, &handle[i], &lp1, &lp2, &p1, &p2, &p3, &p4, buffer);
        if (retcode == ERR_NOMORE) break;
        if (retcode)
        {
            ShowERR(retcode);
            return;
        }

        retcode = Rockey(RY_OPEN, &handle[i], &lp1, &lp2, &p1, &p2, &p3, &p4, buffer);
        if (retcode)
        {
            ShowERR(retcode);
            return;
        }

        i++;
        printf("Find Rock: %08X\n", lp1);
    }
    printf("\n");

    for (j=0;j<i;j++)
    {
        /*
        p1 = 7;
        p2 = 0x2121;
        p3 = 0;

        retcode = Rockey(RY_SET_MOUDLE, &handle[j], &lp1, &lp2, &p1, &p2, &p3, &p4,
buffer);
    }

```

```

if (retcode)
{
    ShowERR(retcode);
    return;
}
printf("Set Moudle 7: Pass = %04X Decrease no allow\n", p2);
p1 = 0;
strcpy((char*)buffer, cmd);
retcode = Rockey(RY_WRITE_ARITHMETIC, &handle[j], &lp1, &lp2, &p1, &p2, &p3,
&p4, buffer);
if (retcode)
{
    ShowERR(retcode);
    return;
}
printf("Write Arithmetic 1\n");
*/
lp1 = 0;
lp2 = 7;
p1 = 1;
p2 = 2;
p3 = 3;
p4 = 4;
retcode = Rockey(RY_CALCULATE1, &handle[j], &lp1, &lp2, &p1, &p2, &p3, &p4,
buffer);
if (retcode)
{
    ShowERR(retcode);
    return;
}
printf("Calculate Input: p1=1, p2=2, p3=3, p4=4\n");
printf("Calculate Output: p1=%x, p2=%x, p3=%x, p4=%x\n", p1, p2, p3, p4);

printf("\n");
printf("Moudle 7 : %x\n", p3);
truelp1=MAKELONG(p2,p1);

printf("truelp1 : %x\n",truelp1);
if (findlp1==truelp1)

    printf("Hello FeiTian!\n");
else
    break;

retcode = Rockey(RY_CLOSE, &handle[j], &lp1, &lp2, &p1, &p2, &p3, &p4, buffer);
if (retcode)

```

```

    {
        ShowERR(retcode);
        return;
    }

    printf("\n");
    getch();
}

}

```

Complex example 2

In Step 28 we get the return codes of a seed code with the calculation 2 function. Then we compare these return codes with the return codes we get with the same seed code at the beginning of the program. If they are different the program will be terminated.

```

#include <windows.h>
#include <stdio.h>
#include <conio.h>
#include "ryvc32.h"

void ShowERR(WORD retcode)
{
    if (retcode == 0) return;
    printf("Error Code: %d\n", retcode);
}

void main()
{
    WORD handle[16], p1, p2, p3, p4, retcode;
    DWORD lp1, lp2;
    BYTE buffer[1024];

    WORD rc[4];

    int i, j;

    char cmd1[] = "A=E|E,B=F|F,C=G|G,D=H|H";

    p1 = 0xc44c;
    p2 = 0xc8f8;
    p3 = 0x0799;
    p4 = 0xc43b;

    retcode = Rockey(RY_FIND, &handle[0], &lp1, &lp2, &p1, &p2, &p3, &p4, buffer);
    if (retcode)

```

```

{
    ShowERR(retcode);
    return;
}
printf("Find Rock: %08X\n", lp1);

retcode = Rockey(RY_OPEN, &handle[0], &lp1, &lp2, &p1, &p2, &p3, &p4, buffer);
if (retcode)
{
    ShowERR(retcode);
    return;
}

i = 1;
while (retcode == 0)
{
    retcode = Rockey(RY_FIND_NEXT, &handle[i], &lp1, &lp2, &p1, &p2, &p3, &p4, buffer);
    if (retcode == ERR_NOMORE) break;
    if (retcode)
    {
        ShowERR(retcode);
        return;
    }

    retcode = Rockey(RY_OPEN, &handle[i], &lp1, &lp2, &p1, &p2, &p3, &p4, buffer);

    if (retcode)

    {
        ShowERR(retcode);
        return;
    }

    i++;

    printf("Find Rock: %08X\n", lp1);
}
printf("\n");
for (j=0;j<i;j++)
{
}

lp2 = 0x12345678;
retcode = Rockey(RY_SEED, &handle[j], &lp1, &lp2, &p1, &p2, &p3, &p4, buffer);
if (retcode)
{
    ShowERR(retcode);
    return;
}

```

```

    }

    printf("Seed: %04X %04X %04X %04X\n", p1, p2, p3, p4);

    rc[0] = p1;
    rc[1] = p2;
    rc[2] = p3;
    rc[3] = p4;

    //  :

    p1 = 0;
    strcpy((char*)buffer, cmd1);
    retcode = Rockey(RY_WRITE_ARITHMETIC, &handle[j], &lp1, &lp2, &p1, &p2, &p3,
&p4, buffer);
    if (retcode)
    {
        ShowERR(retcode);

        return;
    }
    printf("Write Arithmetic 2\n");

    lp1 = 0;
    lp2 = 0x12345678;
    p1 = 1;
    p2 = 2;
    p3 = 3;
    p4 = 4;
    retcode = Rockey(RY_CALCULATE2, &handle[j], &lp1, &lp2, &p1, &p2, &p3, &p4,
buffer);
    if (retcode)
    {
        ShowERR(retcode);
        return;
    }
    printf("Calculate Input: p1=1, p2=2, p3=3, p4=4\n");
    printf("Calculate Output: p1=%x, p2=%x, p3=%x, p4=%x\n", p1, p2, p3, p4);

    printf("\n");
    if(rc[0]==p1 && rc[1]==p2 && rc[2]==p3 && rc[3]==p4)

        printf("Hello FeiTian!\n");
    else
        break;
}

```

```

    retcode = Rockey(RY_CLOSE, &handle[j], &lp1, &lp2, &p1, &p2, &p3, &p4, buffer);
    if (retcode)
    {
        ShowERR(retcode);
        return;
    }

    printf("\n");
    getch();
}
}

```

Complex example 3

In Step 29 we get the values stored in the 16 modules by using the calculation 3 function. Remember that the modules may not be read, even with the Advanced passwords. You may write some important data to the modules or perform some other operations.

```

#include <windows.h>
#include <stdio.h>
#include <conio.h>
#include "ryvc32.h"

void ShowERR(WORD retcode)
{
    if (retcode == 0) return;
    printf("Error Code: %d\n", retcode);
}

void main()
{
    WORD handle[16], p1, p2, p3, p4, retcode;

    DWORD lp1, lp2;
    BYTE buffer[1024];

    int i, j;

    char cmd2[] = "A=E|E,B=F|F,C=G|G,D=H|H";

    p1 = 0xc44c;
    p2 = 0xc8f8;
    p3 = 0x0799;
    p4 = 0xc43b;
}

```

```

retcode = Rockey(RY_FIND, &handle[0], &lp1, &lp2, &p1, &p2, &p3, &p4, buffer);
if (retcode)
{
    ShowERR(retcode);
    return;
}
printf("Find Rock: %08X\n", lp1);

retcode = Rockey(RY_OPEN, &handle[0], &lp1, &lp2, &p1, &p2, &p3, &p4, buffer);
if (retcode)
{
    ShowERR(retcode);
    return;
}

i = 1;
while (retcode == 0)
{
    retcode = Rockey(RY_FIND_NEXT, &handle[i], &lp1, &lp2, &p1, &p2, &p3, &p4, buffer);
    if (retcode == ERR_NOMORE) break;
    if (retcode)
    {
        ShowERR(retcode);
        return;
    }

    retcode = Rockey(RY_OPEN, &handle[i], &lp1, &lp2, &p1, &p2, &p3, &p4, buffer);
    if (retcode)
    {
        ShowERR(retcode);
        return;
    }

    i++;
    printf("Find Rock: %08X\n", lp1);
}
printf("\n");

for (j=0;j<i;j++)
{
    /*
    p1 = 0;
    p2 = 1;
    p3 = 0;
}

```

```

retcode = Rockey(RY_SET_MOUDLE, &handle[j], &lp1, &lp2, &p1, &p2, &p3, &p4, buffer);

if (retcode)
{
    ShowERR(retcode);
    return;
}
printf("Set Moudle 0: Pass = %04X Decrease no allow\n", p2);

p1 = 1;
p2 = 2;
p3 = 0;
retcode = Rockey(RY_SET_MOUDLE, &handle[j], &lp1, &lp2, &p1, &p2, &p3, &p4, buffer);
if (retcode)
{
    ShowERR(retcode);
    return;
}
printf("Set Moudle 1: Pass = %04X Decrease no allow\n", p2);

p1 = 2;
p2 = 3;
p3 = 0;
retcode = Rockey(RY_SET_MOUDLE, &handle[j], &lp1, &lp2, &p1, &p2, &p3, &p4, buffer);
if (retcode)
{
    ShowERR(retcode);
    return;
}
printf("Set Moudle 2: Pass = %04X Decrease no allow\n", p2);

p1 = 3;
p2 = 4;
p3 = 0;
retcode = Rockey(RY_SET_MOUDLE, &handle[j], &lp1, &lp2, &p1, &p2, &p3, &p4, buffer);
if (retcode)
{
    ShowERR(retcode);
    return;
}

printf("Set Moudle 3: Pass = %04X Decrease no allow\n", p2);
// :
*/
p1 = 0;
strcpy((char*)buffer, cmd2);
retcode = Rockey(RY_WRITE_ARITHMETIC, &handle[j], &lp1, &lp2, &p1, &p2, &p3, &p4,

```

```

buffer);
    if (retcode)
    {
        ShowERR(retcode);
        return;
    }
    printf("Write Arithmetic 3\n");

    lp1 = 0;
    lp2 = 0;
    p1 = 0;
    p2 = 0;
    p3 = 0;
    p4 = 0;
    retcode = Rockey(RY_CALCULATE3, &handle[j], &lp1, &lp2, &p1, &p2, &p3, &p4, buffer);
    if (retcode)
    {
        ShowERR(retcode);
        return;
    }
    printf("Calculate Input: p1=0, p2=0, p3=0, p4=0\n");

    printf("\n");
    printf("Moudle 0: %x\n",p1);
    printf("Moudle 1: %x\n",p2);
    printf("Moudle 2: %x\n",p3);
    printf("Moudle 3: %x\n",p4);

    retcode = Rockey(RY_CLOSE, &handle[j], &lp1, &lp2, &p1, &p2, &p3, &p4, buffer);
    if (retcode)
    {

        ShowERR(retcode);
        return;
    }

    printf("\n");
    getch();
}
}

```

Complex example 4

In Step 30 we use all the three calculation functions and we write 4 calculation sections to the ROCKEY dongle. The results of the three calculations are used for additional calculations. Of course you may let ROCKEY perform much more complex calculations according to your situation.

```
#include <windows.h>
#include <stdio.h>
#include <conio.h>
#include "ryvc32.h"

void ShowERR(WORD retcode)
{
    if (retcode == 0) return;
    printf("Error Code: %d\n", retcode);
}

void main()
{
    WORD handle[16], p1, p2, p3, p4, retcode;
    DWORD lp1, lp2;
    BYTE buffer[1024];

    int i, j;
    int t1,t2,t3;

    char cmd[] = "H=H^H, A=A*23, F=B*17, A=A+F, A=A+G, A=A<C, A=A^D, B=B^B, C=C^C,
D=D^D";
    char cmd1[] = "A=A+B, A=A+C, A=A+D, A=A+E, A=A+F, A=A+G, A=A+H";
    char cmd2[] = "A=A+B, A=A+C, A=A+D, A=A+E, A=A+F, A=A+G, A=A+H";
    char cmd3[] = "H=H^H,A=A|A, B=B|B, C=C|C,D=A+B,D=D+C";

    p1 = 0xc44c;
    p2 = 0xc8f8;
    p3 = 0x0799;
    p4 = 0xc43b;

    retcode = Rockey(RY_FIND, &handle[0], &lp1, &lp2, &p1, &p2, &p3, &p4, buffer);
    if (retcode)
    {
        ShowERR(retcode);
        return;
    }
    printf("Find Rock: %08X\n", lp1);

    retcode = Rockey(RY_OPEN, &handle[0], &lp1, &lp2, &p1, &p2, &p3, &p4, buffer);
    if (retcode)
    {
        ShowERR(retcode);
        return;
    }

    i = 1;
```

```

while (retcode == 0)
{
    retcode = Rockey(RY_FIND_NEXT, &handle[i], &lp1, &lp2, &p1, &p2, &p3, &p4, buffer);
    if (retcode == ERR_NOMORE) break;
    if (retcode)
    {
        ShowERR(retcode);
        return;
    }

    retcode = Rockey(RY_OPEN, &handle[i], &lp1, &lp2, &p1, &p2, &p3, &p4, buffer);
    if (retcode)
    {
        ShowERR(retcode);
        return;
    }

    i++;

    printf("Find Rock: %08X\n", lp1);

}
printf("\n");

for (j=0;j<i;j++)
{
    p1 = 7;
    p2 = 0x2121;
    p3 = 0;
    retcode = Rockey(RY_SET_MOUDLE, &handle[j], &lp1, &lp2, &p1, &p2, &p3, &p4,
buffer);
    if (retcode)
    {
        ShowERR(retcode);
        return;
    }
    printf("Set Moudle 7: Pass = %04X Decrease no allow\n", p2);
    printf("\n");

    lp2 = 0x12345678;
    retcode = Rockey(RY_SEED, &handle[j], &lp1, &lp2, &p1, &p2, &p3, &p4, buffer);
    if (retcode)
    {
        ShowERR(retcode);
        return;
    }
    printf("Seed: %04X %04X %04X %04X\n", p1, p2, p3, p4);
}

```

```

printf("\n");

p1 = 0;
p2 = 1;
p3 = 0;
retcode = Rockey(RY_SET_MOUDLE, &handle[j], &lp1, &lp2, &p1, &p2, &p3, &p4,
buffer);
if (retcode)
{
    ShowERR(retcode);
    return;
}
printf("Set Moudle 0: Pass = %04X Decrease no allow\n", p2);

p1 = 1;
p2 = 2;
p3 = 0;
retcode = Rockey(RY_SET_MOUDLE, &handle[j], &lp1, &lp2, &p1, &p2, &p3, &p4,
buffer);
if (retcode)
{
    ShowERR(retcode);
    return;
}
printf("Set Moudle 1: Pass = %04X Decrease no allow\n", p2);

p1 = 2;
p2 = 3;
p3 = 0;
retcode = Rockey(RY_SET_MOUDLE, &handle[j], &lp1, &lp2, &p1, &p2, &p3, &p4,
buffer);
if (retcode)
{
    ShowERR(retcode);
    return;
}
printf("Set Moudle 2: Pass = %04X Decrease no allow\n", p2);

p1 = 3;
p2 = 4;
p3 = 0;
retcode = Rockey(RY_SET_MOUDLE, &handle[j], &lp1, &lp2, &p1, &p2, &p3, &p4,
buffer);
if (retcode)
{
    ShowERR(retcode);
    return;
}

```

```
printf("Set Moudle 3: Pass = %04X Decrease no allow\n", p2);
printf("\n");

p1 = 0;
strcpy((char*)buffer, cmd);
retcode = Rockey(RY_WRITE_ARITHMETIC, &handle[j], &lp1, &lp2, &p1, &p2, &p3,
&p4, buffer);
if (retcode)
{
    ShowERR(retcode);
    return;
}
printf("Write Arithmetic 1\n");

lp1 = 0;
lp2 = 7;
p1 = 5;
p2 = 3;
p3 = 1;
p4 = 0xffff;
retcode = Rockey(RY_CALCULATE1, &handle[j], &lp1, &lp2, &p1, &p2, &p3, &p4,
buffer);
if (retcode)
{
    ShowERR(retcode);
    return;
}
printf("Calculate Input: p1=5, p2=3, p3=1, p4=0xffff\n");

printf("Result = ((5*23 + 3*17 + 0x2121) < 1) ^ 0xffff = 0xBC71\n");
printf("Calculate Output: p1=%x, p2=%x, p3=%x, p4=%x\n", p1, p2, p3, p4);
t1=p1;

p1 = 10;
strcpy((char*)buffer, cmd1);
retcode = Rockey(RY_WRITE_ARITHMETIC, &handle[j], &lp1, &lp2, &p1, &p2, &p3,
&p4, buffer);
if (retcode)
{
    ShowERR(retcode);
    return;
}
printf("Write Arithmetic 2\n");
```

```

lp1 = 10;
lp2 = 0x12345678;
p1 = 1;
p2 = 2;
p3 = 3;
p4 = 4;
retcode = Rockey(RY_CALCULATE2, &handle[j], &lp1, &lp2, &p1, &p2, &p3, &p4,
buffer);
if (retcode)
{
    ShowERR(retcode);
    return;
}
printf("Calculate Input: p1=1, p2=2, p3=3, p4=4\n");

printf("Result =d03a + 94d6 + 96a9 + 7f54 + 1 + 2 + 3 + 4=0x7b17\n");
printf("Calculate Output: p1=%x, p2=%x, p3=%x, p4=%x\n", p1, p2, p3, p4);
t2=p1;

p1 = 17;
strcpy((char*)buffer, cmd2);
retcode = Rockey(RY_WRITE_ARITHMETIC, &handle[j], &lp1, &lp2, &p1, &p2, &p3,
&p4, buffer);
if (retcode)
{
    ShowERR(retcode);
    return;
}
printf("Write Arithmetic 3\n");

lp1 = 17;
lp2 = 0;
p1 = 1;
p2 = 2;
p3 = 3;
p4 = 4;
retcode = Rockey(RY_CALCULATE3, &handle[j], &lp1, &lp2, &p1, &p2, &p3, &p4,
buffer);
if (retcode)
{
    ShowERR(retcode);
    return;
}
printf("Calculate Input: p1=1, p2=2, p3=3, p4=4\n");

```

```

printf("Result = 1+2+3+4+1+2+3+4=0x14\n");
printf("Calculate Output: p1=%x, p2=%x, p3=%x, p4=%x\n", p1, p2, p3, p4);
t3=p1;

printf("\n");
p1 = 24;
strcpy((char*)buffer, cmd3);
retcode = Rockey(RY_WRITE_ARITHMETIC, &handle[j], &lp1, &lp2, &p1, &p2, &p3,
&p4, buffer);
if (retcode)
{
    ShowERR(retcode);
    return;
}
printf("Write Arithmetic \n");

lp1 = 24;
lp2 = 7;
p1 = t1;
p2 = t2;
p3 = t3;
p4 = 0;
retcode = Rockey(RY_CALCULATE1, &handle[j], &lp1, &lp2, &p1, &p2, &p3, &p4,
buffer);
if (retcode)
{
    ShowERR(retcode);
    return;
}

printf("Calculate Output: p1=%x, p2=%x, p3=%x, p4=%x\n", p1, p2, p3, p4);

retcode = Rockey(RY_CLOSE, &handle[j], &lp1, &lp2, &p1, &p2, &p3, &p4, buffer);
if (retcode)
{
    ShowERR(retcode);

    return;
}

printf("\n");
getch();
}
}

```



8.3.3 Advanced Algorithm Application Examples

In Step 31 we will write the core algorithms or codes of the application to the ROCKEY dongle. Below are three programs: the original program, the ROCKEY initializing program and the final program for the end users.

The original program:

```
#include "stdafx.h"
#include "DrawCircle.h"

#include "DrawCircleDoc.h"
#include "DrawCircleView.h"
#include "DrawParamDlg.h"
#include "DrawMethodDlg.h"

#ifndef _DEBUG
#define new DEBUG_NEW
#undef THIS_FILE
static char THIS_FILE[] = __FILE__;
#endif

void CDrawCircleView::DrawCircleMidPoint(CDC *pDC, int iCenterX, int iCenterY, int r)
{
    int x=0;
    int y=r;
    int p=1-r;

    TRACE("Origin\n");

    CirclePlotPoints(pDC,iCenterX,iCenterY,x,y);

    m_lpCircleBuf[0].x = x;
    m_lpCircleBuf[0].y = y;
    m_nPointCount=1;

    while(x<y)
    {
        x++;
        if(p<0)
        {
            p+=2*x+1;
        }
        else
        {
            y--;
            p+=2*(x-y)+1;
        }
    }
}
```

```
        }
        TRACE("%d,(%d,%d);",p,x,y);
        CirclePlotPoints(pDC,iCenterX,iCenterY,x,y);

        m_lpCircleBuff[m_nPointCount].x = x;
        m_lpCircleBuff[m_nPointCount].y = y;
        m_nPointCount++;
    }
    TRACE("\n");
}
```

Initialize ROCKEY:

```
#include "stdafx.h"
#include <windows.h>
#include "..\inc\ryvc32.h"

void ReportErr(WORD wCode)
{
    printf("ERROR:%d\n",wCode);
}

int main(int argc, char* argv[])
{
    WORD p1=0xc44c,p2=0xc8f8,p3=0x0799,p4=0xc43b;
    DWORD lp1,lp2;
    WORD handle[16];
    BYTE buffer[1024];
    BYTE cmdstr[] = "B=B|B,B=B+1,B=B*2,B=B+1,A=A+B,C=C-1,C=C*2,B=A-C";
    WORD retcode;

    retcode = Rockey(RY_FIND,&handle[0],&lp1,&lp2,&p1,&p2,&p3,&p4,buffer);
    if(retcode)
    {
        ReportErr(retcode);
        return 0;
    }
    printf("Find successfully\n");

    retcode = Rockey(RY_OPEN,&handle[0],&lp1,&lp2,&p1,&p2,&p3,&p4,buffer);
    if(retcode)
    {
        ReportErr(retcode);
        return 0;
    }
    printf("Open successfully\n");
```

```
p1 = 10;
retcode = Rockey(RY_WRITE_ARITHMETIC,&handle[0],&lp1,&lp2,&p1,&p2,&p3,&p4,cmdstr);
if(retcode)
{
    ReportErr(retcode);
    return 0;
}
printf("Write arithmetic successfully\n");

retcode = Rockey(RY_CLOSE,&handle[0],&lp1,&lp2,&p1,&p2,&p3,&p4,buffer);

return 0;
}
```

The final program for the end users:

```
#include "stdafx.h"
#include "DrawCircle.h"

#include "DrawCircleDoc.h"
#include "DrawCircleView.h"
#include "DrawParamDlg.h"
#include "DrawMethodDlg.h"

#ifndef _DEBUG
#define new DEBUG_NEW
#undef THIS_FILE
static char THIS_FILE[] = __FILE__;
#endif

WORD p1=0xc44c,p2=0xc8f8,p3=0x0799,p4=0xc43b;
DWORD lp1,lp2;
WORD handle[16];
BYTE buffer[1024];

void CDrawCircleView::DrawCircleMidPoint_Rockey(CDC *pDC, int iCenterX, int iCenterY, int r)
{
    int x=0;
    int y=r;
    int p=1-r;
    int seed=0;

    short p1,p2,p3,p4;

    CirclePlotPoints(pDC,iCenterX,iCenterY,x,y);

    TRACE("Hardware\n");
    m_lpCircleBuf[0].x = x;
```

```
m_lpCircleBuf[0].y = y;
m_nPointCount=1;

while(x<y)
{
    p1 = p;
    p2 = x;
    p3 = y;
    p4 = seed;

    if(!RunRockey((WORD&)p1,(WORD&)p2,(WORD&)p3,(WORD&)p4))
    {
        //      AfxMessageBox("Error during run time");
        break;
    }

    if(p<0)
    {
        p = p1;
    }
    else
    {
        p = p2;
        y--;
    }
    x++;
    TRACE("%d,(%d,%d);",p,x,y);
    CirclePlotPoints(pDC,iCenterX,iCenterY,x,y);

    m_lpCircleBuff[m_nPointCount].x = x;
    m_lpCircleBuf[m_nPointCount].y = y;
    m_nPointCount++;
}
TRACE("\n");
}

BOOL CDrawCircleView::RunRockey(WORD &A, WORD &B, WORD &C, WORD &D)
{
    WORD retcode;

    lp1 = 10;
    retcode = Rockey(RY_CALCULATE1,&handle[0],&lp1,&lp2,&A,&B,&C,&D,buffer);

    if(retcode)
        return FALSE;
    else
        return TRUE;
}
```



8.4 Note

The standard ROCKEY4 dongle has a 32 WORD instruction space so it can accommodate at most 32 instructions. The NetROCKEY4 and ROCKEY4-Plus dongles can each accommodate as many as 80 instructions. Developers do not need to consider the start and end attributes of an algorithm. ROCKEY will automatically assign a Start/End attribute to the instructions. In practice this means that if the developer writes a two-instruction algorithm to the User Algorithm Zone (UAZ), and then a three instruction algorithm, the result will not be a single five instruction algorithm. Algorithms that begin with “ Null” or “ E” will produce unpredictable results.

8.5 Tips

1. **Make randomized calls to the ROCKEY API** - Randomly scatter calls to the ROCKEY API from within your application. Calls made to the API from time-to-time will make it very difficult to mimic the behavior of the protection method or hack the application.
2. **Use dynamic information with the seed code function** - The use of dynamic information with the seed code function, such as system date, makes the protection method stronger because the results can change with the input and calculation.
3. **Do not repeatedly use the same protection method in your application** - If you use the same protection method several times in your application it will be easier for the cracker to find the rule and crack your application. Protection methods that are complex and rely on a number of different checks and calculations are the most difficult to crack.
4. **Encrypt the character string and data** – In “Step 18” of this document we showed an encryption method using information stored inside the dongle. Encrypting a character string in the manner described is a strong method because a failure to properly decrypt the string can cause the application to terminate or take other action in accordance with the licensing agreement.
5. **Use API encryption and Envelop encryption together** – The strongest protection method will have the developer first using a complex and dynamic implementation of the ROCKEY API, and then protecting this new file with the ROCKEY Envelop.

Please keep the end user environment in mind when you design the software protection solution. You should flexibly adopt the methods suggested here within the limitations and objectives of your environment and licensing policy.

Chapter 9

NetROCKEY4

NetROCKEY4 is a network aware software protection system designed to limit the number of simultaneous users who can access a software application. It combines all the functionality of the standard ROCKEY4 system with the ability to work seamlessly in LAN/WAN environments that support the UDP/TCP, IPX or NetBIOS protocols.

NetROCKEY4 was engineered to support standalone or redundant server environments on both the Windows and Linux platforms. The system includes powerful and intuitive network monitoring and testing tools that ease the implementation effort. It also offers a brand new NetROCKEY Envelop encryption tool and enhanced algorithm zone protection mechanism.

9.1 NetROCKEY4 Basic Concepts

1. **Configuration Files** - There is a configuration file for the Service program (*svrcfg.ini*), and one for the Client program (*clicfg.ini*). The service and client programs take their network settings from the configuration files. The developer may use a text editor or tools provided by Feitian (*NrConfig.exe* under directory *Net\Tools\NrConfig*) to edit such configuration file parameters as: protocol type, time -to-live, server address and other information required for the dongle to attach to the network.
2. **Log Files** - The NetROCKEY4 log file (*svrlog.txt*) records the running status of the service program. It can be helpful if you encounter a problem with the service program. The path and name of the log file may be configured in the *svrcfg.ini* file.
3. **Port and Group Information** - The UDP/TCP and IPX protocols require the specification of a port number. Port numbers range from 0 to 65535. The default port number for NetROCKEY4 is 3152. 3152 is registered with IASA and should be available on most networks. If it is occupied though, the NetROCKEY4 service program will report a “bind” error. If a bind error occurs you may want to move NetROCKEY4 to an available port; the port number can be changed in the *svrcfg.ini* file. The NetBIOS protocol does not use a port number. It uses a group name. The group name is a character string that may be a maximum of 16 characters in length. Each server in a NetBIOS network has both a computer name and a group name. The group name for the service program may be altered in the *svrcfg.ini* file and the group name for the clients in the *clicfg.ini* file. All clients and servers that need to communicate in a NetBIOS network need to have the same group name.



4. **Network Address** - Each computer in network has a unique address. A UDP/TCP network (IPv4) uses IP addresses like xxx.xxx.xxx.xxx (such as 192.168.0.1). An IPX network uses a 6-byte MAC address that may look like: 00-35-4f-20-00-32. A NetBIOS network uses a computer name (16 characters).
5. **Search Mode** - The NetROCKEY4 client program will search for the address of the service program at start-up together with the protocol type and port in the configuration file. The client program will look to the *clicfg.ini* file for the search “ mode” . There are three search modes that may be set in the client configuration file: Automatic, Custom and Semi-automatic. Automatic mode means that the client will broadcast to locate all the service programs on the network. Custom mode requires that you enter a search list of the service addresses in the *clicfg.ini* file. The client program will not issue a broadcast message but will use the search list to find the service programs. Automatic mode has the advantage of being easy to configure, but the drawbacks of slow response and added network overhead. Custom mode is faster than automatic but requires that you know the addresses of the service programs. Semi-automatic mode attempts to overcome the drawbacks of both the custom and automatic modes. In semi-automatic mode, the client will first go to its search list. If it finds one or more service programs it will quit searching. If it does not find a service program, it will broadcast to find the service program.
6. **Open Mode** - The NetROCKEY4 client programs issue an “open” command to the service program. This open command is equivalent to a network login and it is the means by which the service program limits the number of users that can attach to the application. There are two operating modes for the open command: private and share. The default setting is private mode. In private mode operation the service program adds “ 1” each time a user attaches to the application. If the calculated quantity reaches the maximum set by the developer, the open command will fail, the service program will issue an error message and the user will not be allowed to access the application. In share mode, all programs in the same computer share one user number. No matter how many times the computer accesses the service program, it is considered to be one user. Share mode is appropriate if the number of computers, rather than the number of users that attach to an application, need to be limited. The open mode is set with the lp2 parameter in the open operation. The low byte of the lp2 parameter sets the NetROCKEY4 module number that will store the maximum number of simultaneous users, and the high byte sets the open mode. (Please refer to NetROCKEY API Services)
7. **Time Out** - Each time the client sends data to the server it will wait a time period defined by the “time out” parameter. If the client does not receive a response after the time out period, it will quit and return an error code. The unit of time for the time out parameter is seconds and the default is two. In automatic search mode, the time out is also the period that the client program will wait for a response to its broadcast message. The time out parameter can be changed in the *clicfg.ini* file.
8. **Maximum Number of Simultaneous Users** - The maximum number of simultaneous users that will be allowed to access an application will be set by a value stored in a NetROCKEY4 module. For example, if you write “ 5” to module number 0, module 0 can be used to set a limit of five users who can simultaneously use your application. A NetROCKEY4 dongle has 16 modules, so



as many as 16 user groups can have individualized application access limits. The NetROCKEY4 Editor program can be used to write a value to a module, but it is impossible to read the value.

9. **Client Time To Live (TTL)** - This parameter is set in the service program (*svrcfg.ini*). The time unit is minutes and the default is 3. The client program automatically sends an “idle” message to the service program once each 1.5 minutes. If the service program does not receive an idle message from a client for the TTL period, it will delete the client handle, terminating the connection. Thus in the event the client is shutdown abnormally or the network connection is lost one client cannot be considered as two or more users by the service program.
10. **Open Module** – A module can only be opened once in a single process with security in mind. You may set the handle as a global variable to use it in every thread.

9.2 NetROCKEY4 Developer’s Kit

The development tools for NetROCKEY in directory “Net”:

<Client> DLLs and configuration files for the NetROCKEY4 client program
<Server> Executable and configuration files for the NetROCKEY4 service program.
<Tools> Development tools for NetROCKEY4
<Samples> Sample program files

The sections below will discuss the main functions of these tools.

9.3 NetROCKEY4 Configuration Files and Tools

The configuration tool is in directory <NrConfig>. Its files are listed below:

NrConfig.exe Configuration file editor to control service and client programs.
svrcfg.ini Configuration file for the NetROCKEY4 service program.
clicfg.ini Configuration file for the NetROCKEY4 client program.
NrClient.dll API library for the NetROCKEY4 client program (You should not change the name of this file).

There is a configuration file for the Service program (*svrcfg.ini*), and one for the Client program (*clicfg.ini*). The configuration files will configure the settings of the network (All the characters in the configuration files are case sensitive). Below is the template of Client Program Configuration File (*clicfg.ini*):

```
[Header]
Sign = RockeyClientHeader
      ; Client configuration file.
```

[Common]
Timeout =2
;Time out value. Unit: seconds. Default: 2.

SearchFlag =0
; Search mode. 0=Automatic mode, 1=Custom mode, 2=Semi-automatic mode.
; If you set Custom mode, you must enter a search list for each protocol.

[TCPUDP]
bUseTCP =1
bUseUDP =1
; Enable TCP/ UDP protocol. 1=Yes, 0=No.
TCPPort =3152
; TCP port. Must be the same as the server.
; 3152 is a registered TCP port for NetROCKEY4.
UDPPort =3152
; UDP port. Must be the same as the server.
; 3152 is a registered UDP port for NetROCKEY4.

SearchList =192.168.0.16, 192.168.0.1,wenlong
; Search list. Used in custom or semi-automatic mode.
; IP addresses and server names need to be separated by a “,”.

[IPX]
bUsed =0
; Enable IPX protocol. 1=Yes, 0=No.
IPXPort =3152
; IPX port. Must be the same as the server.
SearchList =00-A0-0C-13-0E-D2, 00-00-B4-B2-ED-7B
; Search list. Used in custom or semi-automatic mode.
; MAC addresses need to be separated by a “,”.
; MAC address can be obtained with the command “nbtstat -a pc name” .
; You may use our configuration program (*NrConfig.exe*) to change the computer name to
; MAC address.

[NetBios]
bUsed =0
; Enable NetBIOS protocol. 1=Yes, 0=No.
RegGrpName =FTNetServer
; Group name of servers. Default setting is FTNetServer.
; It must be the same as the name in the *svrcfg.ini* file.

SearchList=book,wenlong
; Search list. Used in custom or semi-automatic mode.
; Server names need to be separated by a “,”.
; The server name may be the computer name from the operating system or the name entered
; in the *svrcfg.ini* file.

Service Program Configuration File (*svrcfg.ini*) - Template

[Header]

Sign=RockeySvrHeader

; Service program configuration file.

[Common]

Timeout=2

; Time out value. Unit: seconds. Default: 2.

IdleTime=3

; Time To Live (TTL) value. Unit: minutes. Default: 3.

; Client program sends idle message to service program every 1.5 minutes. The service

; program will kill a client if it does not receive an idle message from the client a time

; interval set here. This parameter applies to a situation in which the client shuts down his
; computer or quits before closing NetROCKEY4.

LogFile=svrlog.txt

; Name and path for the log file.

; The log file records information output by the service program.

[TCPUDP]

bUsed=1

; Enable TCP/UDP protocol. 1=Yes, 0=No.

TCPPort=3152

; TCP port. Default is 3152. 3152 is a registered port address.

; If this port is already used in your network, you may change it here.

UDPPort=3152

; UDP port. Default is 3152. 3152 is a registered port address.

; If this port is already used in your network, you may change it here.

[IPX]

bUsed=0

; Enable IPX protocol. 1=Yes, 0=No.

IPXPort=3152

; IPX port. Default setting is 3152.

; If this port is already used in your network, you may change it here.

[NetBios]

bUsed=0

; Enable NetBIOS protocol. 1=Yes, 0=No.

RegName=FTNetServer

; Register server name. Default: FTNetServer001. You may enter a new name. If the server

; name already exists in your network, you may increment the number at the end of the name,

; for example: FTNetServer002, FTNetServer003...

RegGrpName=FTNetServer

; Group name for servers. Default: FTNetServer. You may change the group name here.

; The service configuration file and client configuration file must be consistent.

The Configuration File Editor is a graphical program that may be used to edit *svrcfg.ini* and *clicfg.ini*. The screen pictured in Figure 9.1 will appear if neither *svrcfg.ini* nor *clicfg.ini* is found in the current directory. Click on either or both of the check boxes to create the configuration file(s) in the current directory with default settings.

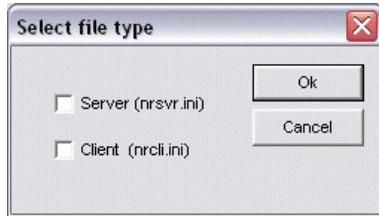


Figure 9.1

The editor may edit *svrcfg.ini* and *clicfg.ini* files in the current directory, the screen is pictured below:

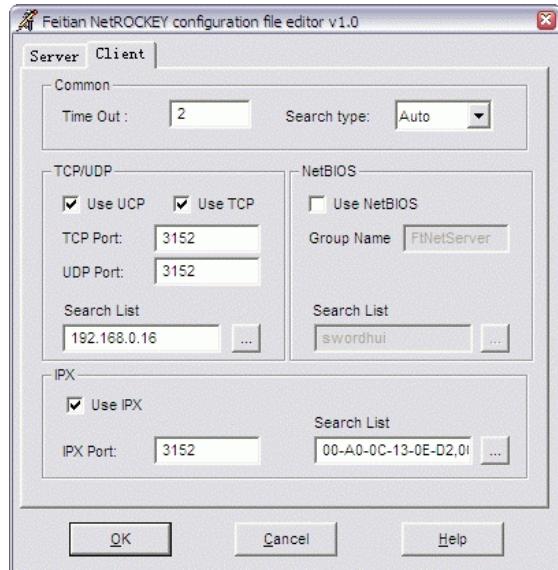


Figure 9.2

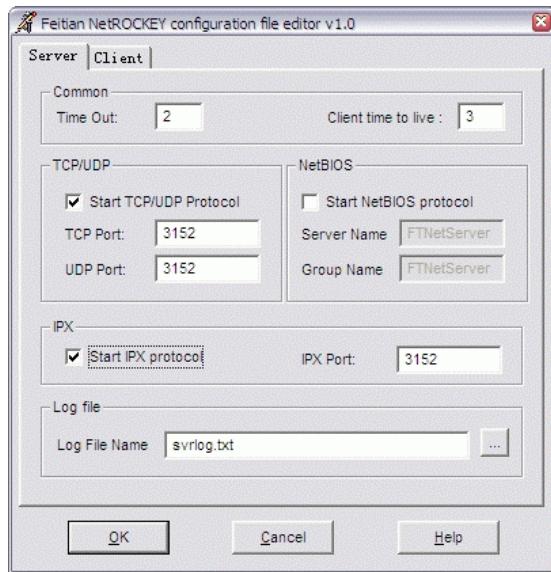


Figure 9.3

Hold the mouse pointer on a particular field for a couple of seconds for a helpful tip. An Editor screen with a “tip” caption is shown in Figure 9.4:

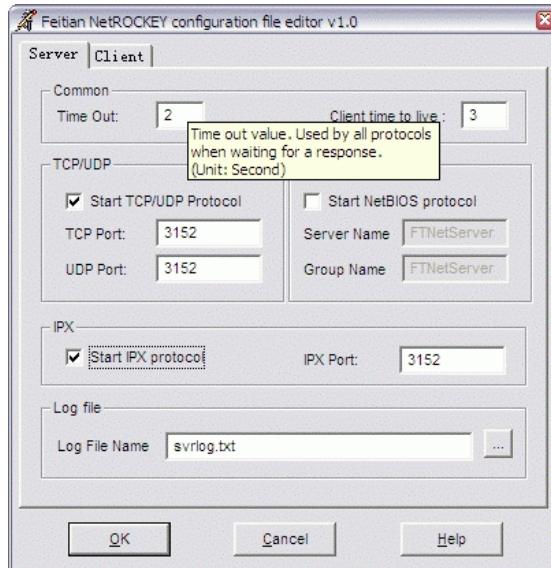


Figure 9.4

Note: If only *svrcfg.ini* or *clicfg.ini* is in the current directory, the Configuration file Editor will allow you to edit the file that it finds. You can also extract the other *.ini* file by clicking on the icon in the upper left portion of the screen. A pop-up menu will then appear that will allow you extract the file you need.

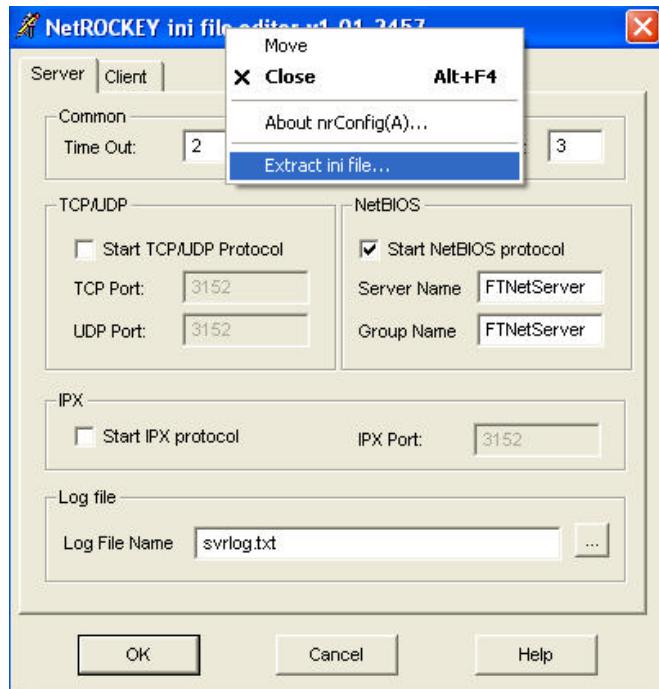


Figure 9.5

9.4 NetROCKEY4 Service Program

The service program (*NrSvr.exe*) is under directory <Server>. NetROCKEY is detected by the client program only after the service program is started on the computer to which the NetROCKEY is attached.

After the service program is run for the first time it will automatically register itself as the service program. It will run automatically every time the computer is started unless it is uninstalled. After it is started the service program will look in the current directory for the service configuration file (*svrcfg.ini*) and take the configuration information. If it can not find the configuration file it will use the default configuration. The status of the Service program will be recorded in a log file specified in the *svrcfg.ini* file.

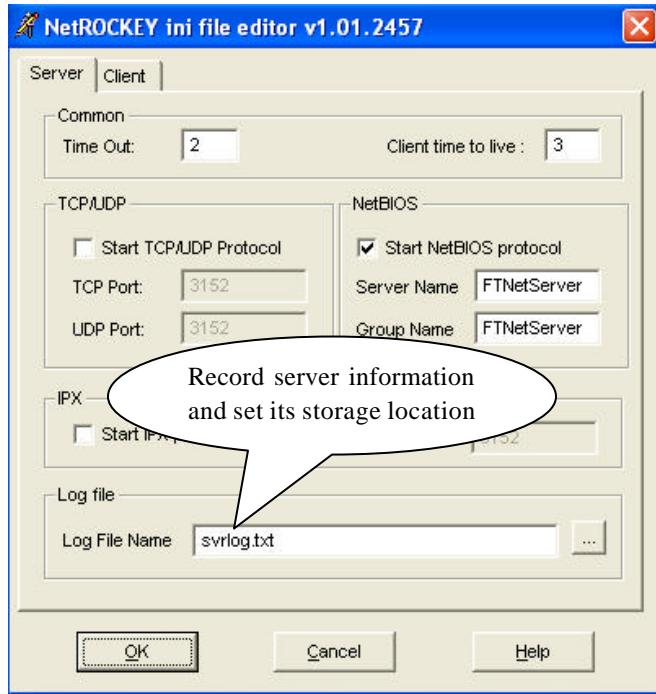


Figure 9.6

An icon will appear in the system tray when the service program is started. See Figure 9.7

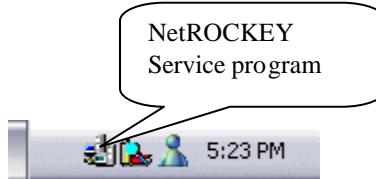


Figure 9.7

Double click or right click the Service program icon to open the screen pictured in Figure 9.8.



Figure 9.8

The FT Service Control screen may be used to stop, start or uninstall the Service program. Right click the Service program icon to open a menu to start, stop, uninstall or exit the program.

Note: The Service program requires a driver and a NetROCKEY but client program does not.

9.5 NetROCKEY4 Monitor

The monitor program (*NrMon.exe*) is under the directory *Net\Tools\NrMon*, it will function on any LAN attached PC, it does not require a ROCKEY4 driver and Dll. It was designed to monitor the activities of all NetROCKEY4 devices on the network. If it is installed on the PC running the Service program, it can also be used to start and stop the Service functions or kill a client. If *svrcfg.ini* or *clicfg.ini* is under the executing directory of *NrMon.exe*, *NrMon.exe* will automatically take the port information of these files to connect the network.

The Monitor program will first search for all network attached service programs and clients, as shown in Figure 9.9. You can specify the protocol for the search operation. Select the search protocol from the “Setting” pull-down menu or from the tool bar. Click the button to invoke the application.

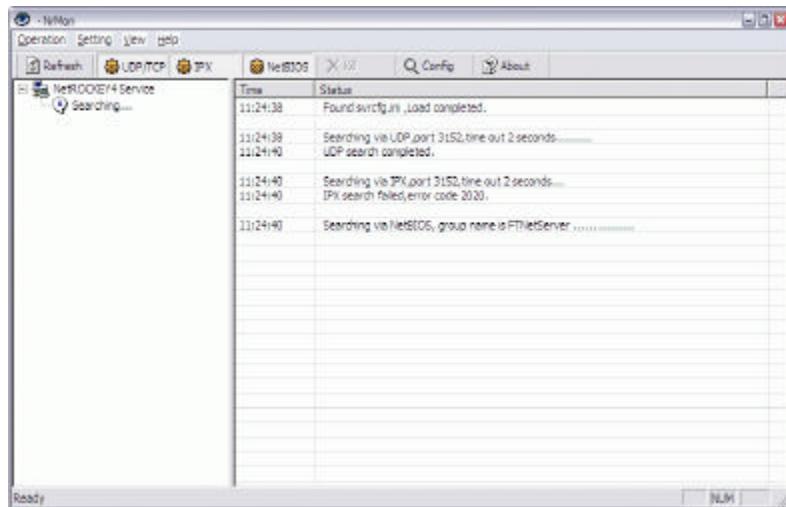


Figure 9.9

Return code will appear if there is an error reported by a protocol and you may refer to the Return Codes section to find out the reason. The search results will be displayed in Figure 9.10.

The server names appear on the left portion of the window. The NetROCKEY4 hardware ID (HID) will appear if users are logged into the server. Server information, including server platforms and opened protocols, appears in the right portion of the screen.

If the Monitor program is installed on the same computer as the service program, the word “ Local” will appear next to the computer name , and you may control the service via monitor, including starting and stopping the protocols or killing a client. You may also invoke these operations from operation menu or tool bar.

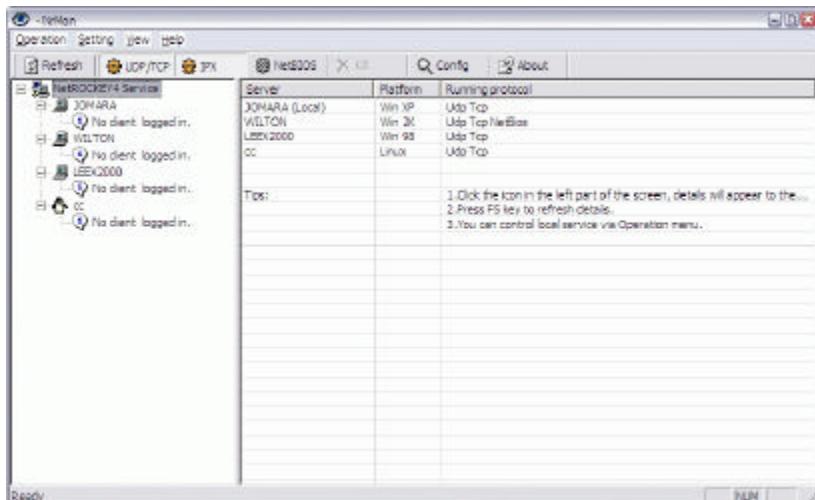


Figure 9.10

In Figure 9.11 the user has selected the HID of a server.

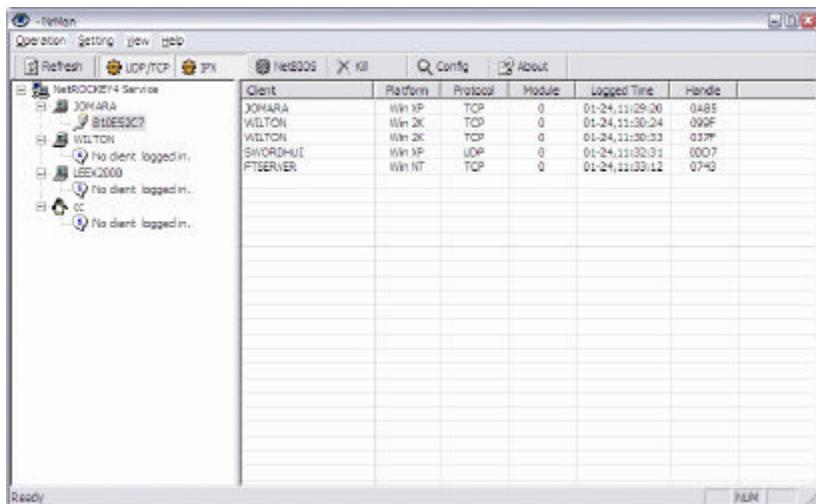


Figure 9.11

Client information, including computer name, platform, opened protocol, log-in module, log-in time, and handle, appears in the right portion of the screen. If a user logs into the local server you can delete the connection by clicking on the client and then pressing the “Delete” key, or clicking the “Kill” button on the toolbar.

ROCKEY

Pressing F5 key or choosing the “ Refresh” button on the toolbar can refresh the current screen. Auto-refresh mode may be activated from the toolbar or pull-down menu. Please see the Auto-Refresh screen pictured below:



Figure 9.12

Simply click the “Auto Refresh” option, enter a time interval (seconds) and then click “OK”. Press Refresh button or Auto Refresh only refreshed the current screen. Click root, server or NetROCKEY in the left part of the screen to display and refresh the corresponding information - click the root to refresh all the service information, click a server to refresh its server information or click a NetROCKEY to refresh its client information.

You can stop or start specific protocols from the “ Operation” pull down menu. Protocols can only be stopped or started from the Monitor program if it is running on the same machine as the Service program.

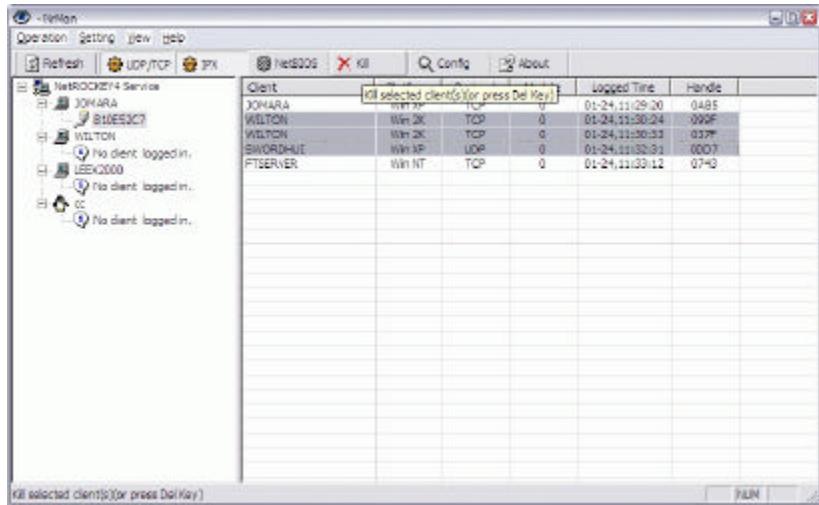


Figure 9.13

Choose the clients logged on local computer, move the mouse pointer to Kill button, the button will brighten. Clicking the “Kill” button on the toolbar or pressing the “Delete” key will delete the connection to the selected clients. See Figure 9.13 above.

Note: The server icon will be a penguin if the service platform is Linux. See Figure 9.14.

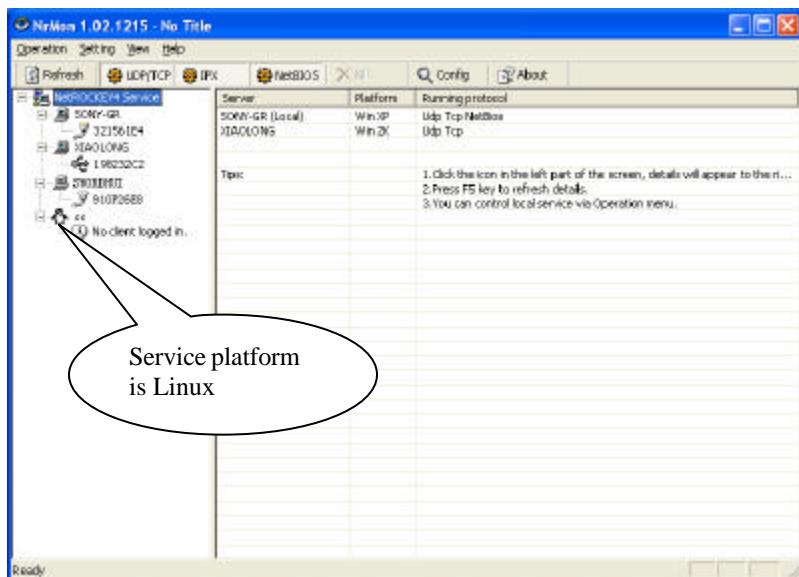


Figure 9.14

9.6 NetROCKEY4 Test Utility

Test utility (*NrTest.exe*) is under the directory *Net\Tools\NrTest*. It was designed to test the functions of NetROCKEY4. *NrTest.exe* will take the client configuration file in current directory to test NetROCKEY4 system. It requires both *NrClient.dll* and *clcfg.ini* in the current directory but it does not require the ROCKEY4 driver.

The NetROCKEY4 Test utility requires password access. See Figure 9.15 below.

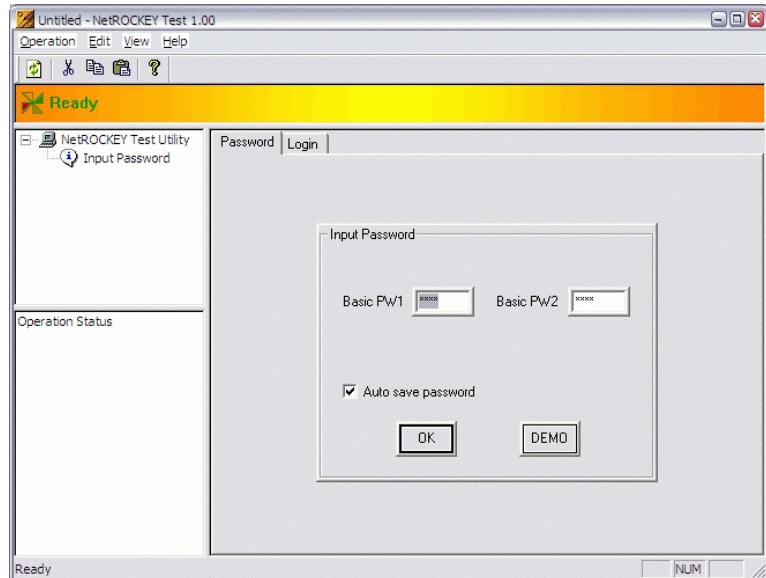


Figure 9.15

The Basic passwords alone allow full functionality of the Test program. NetROCKEY4 server does not accept Write operation with Advanced passwords. If you need to change the content of NetROCKEY4, you must edit the content of it with the ROCKEY4 Editor program. The “Auto save password” option will encrypt the entered passwords and save them in the system registry. This feature is handy if you do not want to reenter the passwords each time you work with the Test utility. If you are working with a Demo NetROCKEY4 dongle, click the “DEMO” button, no password entry is required. The next step is to search for NetROCKEY4 network clients. This action requires the *NrClient.dll* and *clicfg.ini* files. The Test utility will show error information if *NrClient.dll* is not found. See Figure 9.16.

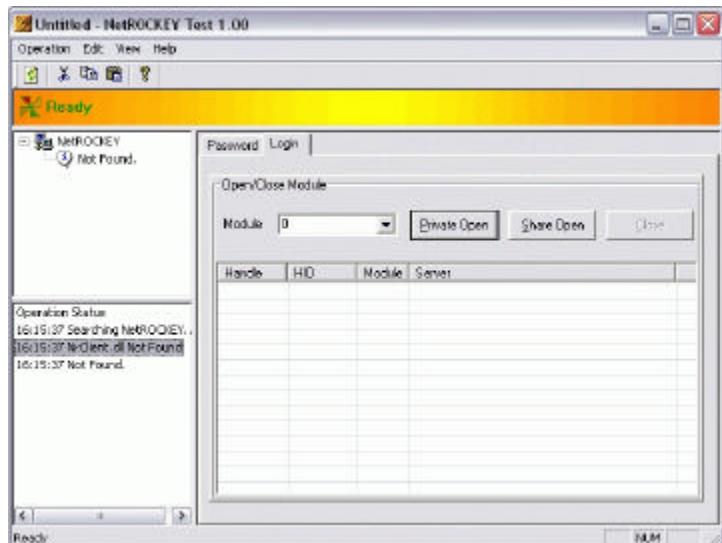


Figure 9.16

If this problem occurs, copy the *NrClient.dll* and *clicfg.ini* files to current directory. The test utility will automatically take network configuration information, such as protocol and ports, found in file *clicfg.ini* and search with the function in *NrClient.dll*. If *clicfg.ini* is not found, the system will use the default configuration. The searching screen is pictured below:

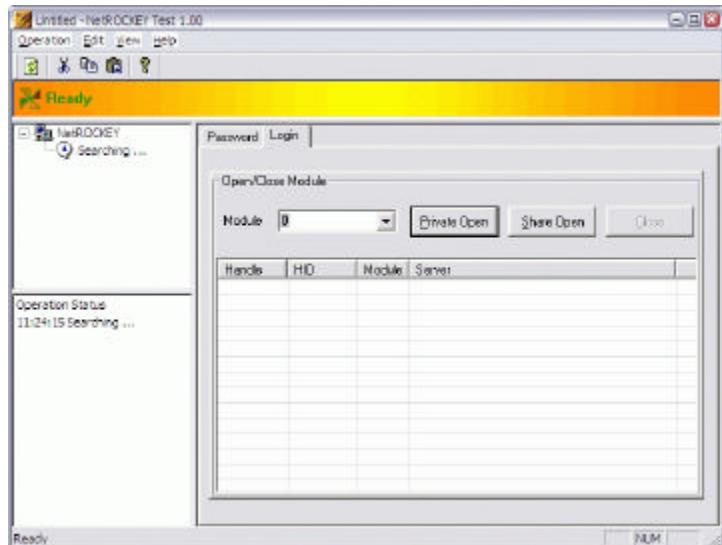


Figure 9.17

ROCKEY

The results of a successful search will look like Figure 9.18, all server and NetROCKEY are displayed here.

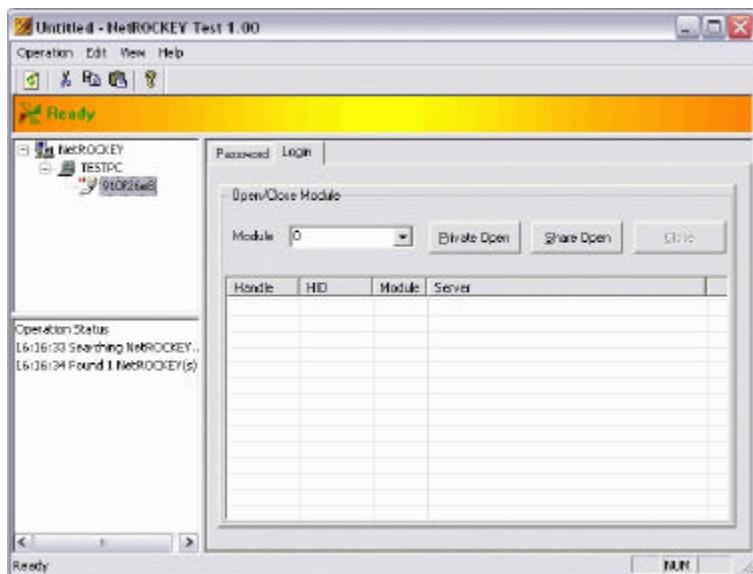


Figure 9.18

Select a NetROCKEY and the log-in screen will appear. Choose the module you would like to open and click either the “ Private Open” or “ Share Open” . You will get a return handle for future operations. See Figure 9.19.

ROCKEY

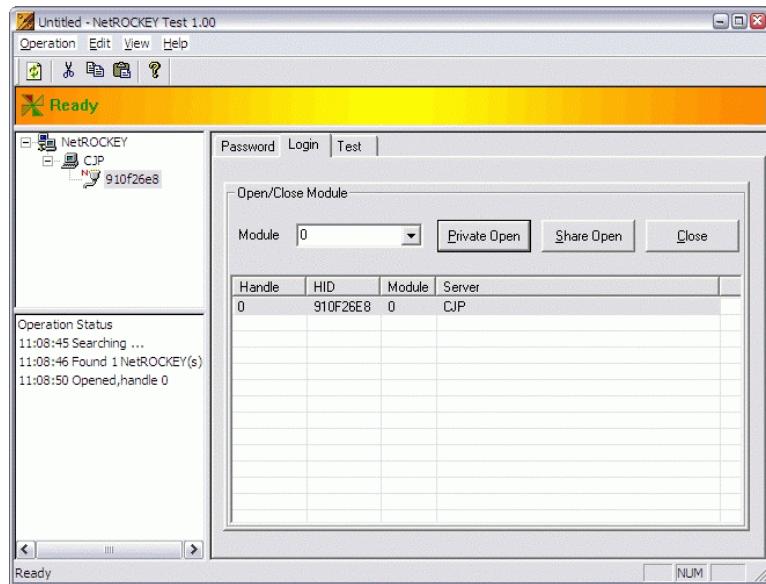


Figure 9.19

Here the handle is returned. You may click the “Test” tab to operate on the returned handle. See Figure 9.20.

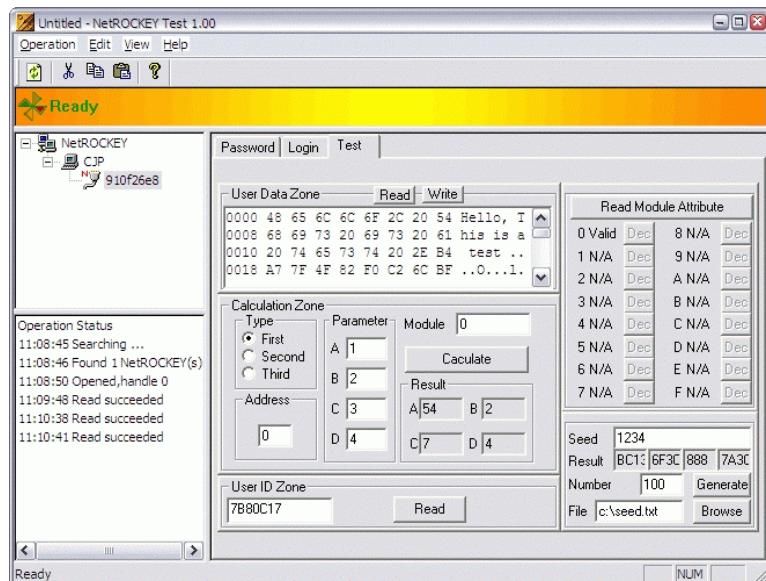


Figure 9.20

You may test the functions: read/write to UDZ, read user ID and the status of the module, calculate and generate seed random number.

All input must be in hexadecimal with the exception of the seed generation number and character string. After the test please return to the log-in screen to close the handle. If you do not close the handle the service program will kill this client several minutes later (refer to Time To Live). The Test and Monitor programs together can be used to verify and trouble shoot the status of the entire NetROCKEY4 system.

9.7 NetROCKEY4 Envelop Encryption

The envelop tool (*Shell.exe*) is under the directory Net\Tools\Shell. It is an ideal solution for those who do not have the source code or cannot call the API. The encrypted program can run independently. It does not require the ROCKEY4 driver or *NrClient.dll*. It only requires that *clicfg.ini* be installed on the client machine. Its graphical interface makes it very easy to encrypt 32-bit Windows PE files (i.e. exe and dll files). See Figure 9.21.

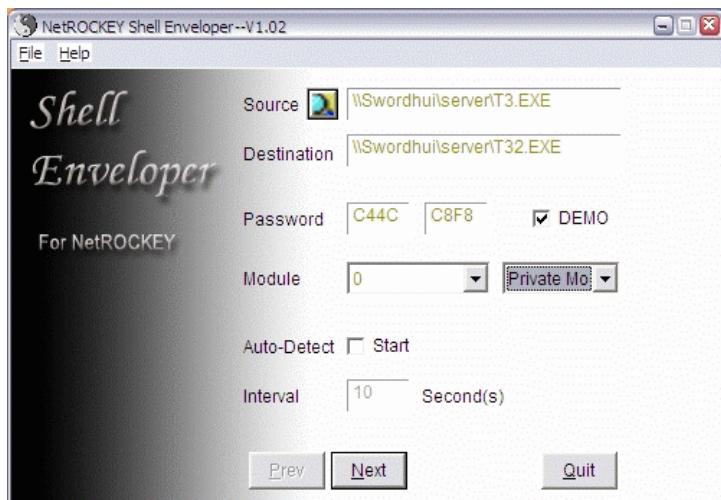


Figure 9.21

Enter the source and destination file names in the first two edit boxes. We suggest you do not overwrite the source file – assign a new name to the destination file. Enter the Basic passwords or choose DEMO if it is a demo dongle. Then choose the module and log-in mode. Set the time interval if you activated the Auto-Detect function. The program that is running will be automatically terminated if the NetROCKEY dongle is unplugged, the service is closed or the service program is malfunctioning – unsaved data may be lost.

NetROCKEY must be attached to the computer that runs the Envelop encryption function and drivers are required. A dialogue box will appear if one of these components are missing.

Click “Next” . You may edit the error message in the screen below:

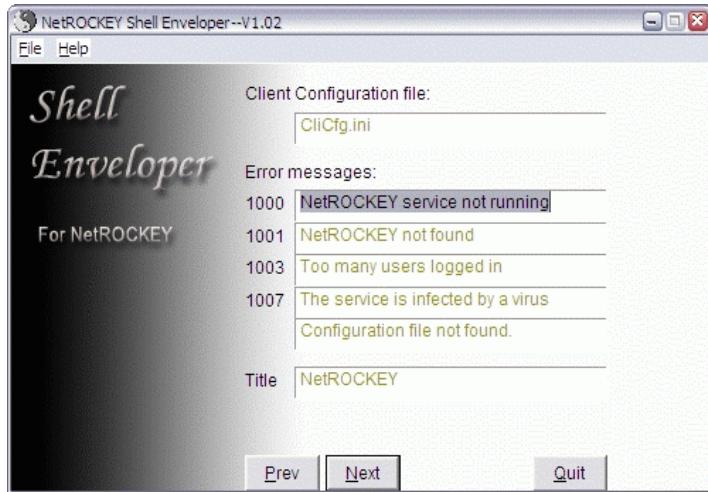


Figure 9.22

Edit the error message and specify the client configuration file. The program will open the specified configuration file and connect the network at the next start-up. The edited dialogue box will appear if there is an error.

Choose “Next” to perform the envelop encryption function. If it succeeds a screen similar to below will appear:

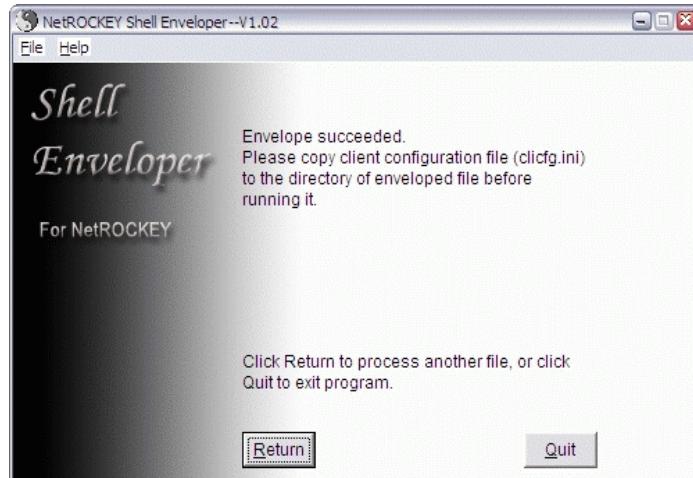


Figure 9.23

Press “Quit” to exit the program or “Return” to encrypt another program. If the action fails a message will display to indicate the reason.

An error message like the one below will appear if a user attempts to run an Envelop protected program without a NetROCKEY dongle attached to the PC.

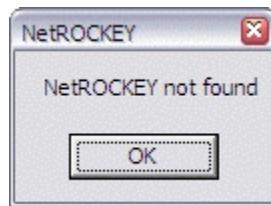


Figure 9.24

9.8 NetROCKEY4 Function Prototype and Definition

9.8.1 NetROCKEY4 Function Prototype

NrClient.dll supports NetROCKEY4 function call, the 4 functions are below:
DWORD WINAPI NetRockey

```
(  
    WORD      function,  
    WORD      *handle,  
    DWORD     *lp1,
```

```
        DWORD      *lp2,
        WORD       *p1,
        WORD       *p2,
        WORD       *p3,
        WORD       *p4,
        BYTE       *buffer
    );
DWORD WINAPI SetIniPathName(LPCTSTR iniName);
DWORD WINAPI NrGetLastError();
DWORD WINAPI NrGetVersion();
```

NetRockey is the main function for NetROCKEY4.

9.8.2 Parameters

NetRockey provides NetROCKEY4 function call. A return code of “0” indicates the operation succeeded. All other return codes indicate an error. Please refer to section on return codes. A function is a 16-bit number and it indicates the specific function of NetRockey. They are defined below:

#define RY_FIND	1	Find NetROCKEY4
#define RY_FIND_NEXT	2	Find next NetROCKEY4
#define RY_OPEN	3	Open NetROCKEY4
#define RY_CLOSE	4	Close NetROCKEY4
#define RY_READ	5	Read NetROCKEY4
#define RY_WRITE	6	Write NetROCKEY4
#define RY_RANDOM	7	Generate Random Number
#define RY_SEED	8	Generate Seed Code
[*]#define RY_WRITE_USERID	9	Write User ID
#define RY_READ_USERID	10	Read User ID
[*] #define RY_SET_MOUDLE	11	Set Module
#define RY_CHECK_MOUDLE	12	Check Module
[*] #define RY_WRITE_ARITHMETIC	13	Write Algorithm
#define RY_CALCULATE1	14	Calculate 1
#define RY_CALCULATE2	15	Calculate 2
#define RY_CALCULATE3	16	Calculate 3
#define RY_DECREASE	17	Decrease Module Unit

Note: Function Parameters 9, 11 and 13 are not valid for NetROCKEY4 because they require the Advanced passwords. The Service program will not recognize these parameters. The Editor for the standard (standalone) ROCKEY may be used to update these functions in the NetROCKEY product. These NetROCKEY functions only require Basic passwords.



Handle	NetROCKEY4's handle	(16 bit)
lp1,lp2	long integer parameter	(32 bit)
p1,p2,p3,p4 buffer	short integer parameter Buffer	(16 bit)

Please refer to the API section for more detail.

SetIniPathName - Set the pathname of a client configuration file. A return code of “ 0” indicates the operation succeeded. All other return codes indicate an error. IniName is the name of file, such as “c:\clcfg.ini”.

NrGetLastError - Get the last error code from NetROCKEY4. This function will also return the error code description. Please refer to section Return Code to see a list of error codes.

NrGetVersion - Get the version number of *NrClient.dll*. The high WORD of return value is the main version number and the low WORD is the secondary version number.

9.9 NetROCKEY4 API Services

The NetROCKEY4 API function parameters are defined in detail below. The password 3 and 4 (Advanced passwords) should be set to “0” in the final product delivered to end users. The functions marked with [*] require the Advanced passwords.

1. Find a NetROCKEY4 dongle (RY_FIND)

Objective: Find NetROCKEY4 client and service programs according to parameters set in the configuration file.

Input parameters:

function = RY_FIND

*p1 = Password 1

*p2 = Password 2

Return value:

A return value = “ 0” indicates that the function worked correctly. Any other return value indicates an error. A successful operation will write the NetROCKEY4 Hardware ID (HID) to *lp1 and the server name to the buffer.

2. Find the Next NetROCKEY4 dongle (RY_FIND_NEXT)

Objective: To check if another NetROCKEY4 dongle is attached to the network.

**Input parameters:**

function = RY_FIND_NEXT
*p1 = Password 1
*p2 = Password 2
*lp1 = Hardware ID of last found dongle

Return value:

A return value = “ 0” indicates that the function worked correctly. Any other return value indicates an error. A successful operation will write the NetROCKEY4 Hardware ID (HID) to *lp1 and the server name to the buffer.

3. Open the NetROCKEY4 dongle (RY_OPEN)

Objective : To log into a specified NetROCKEY4 module and get a handle number to enable other operations.

Input parameters:

function = RY_OPEN
*p1 = Password 1
*p2 = Password 2
*lp1 = Hardware ID
*lp2 = High word is open mode (0 = private mode, 1 = share mode).
Low word is module number (0 to 15).

For example, *lp2=1 means log in module 1 in private mode. *lp2 = 0x10001 means log in module 1 in share mode. You can also use the VC MAKELPARAM macro, such as lp2 = MAKELPARAM(1,1);

Return value:

A return value = “ 0” indicates that the function worked correctly. Any other return value indicates an error. A successful operation will return the correct NetROCKEY4 handle.

Note: You must use the same *p1, *p2 values with the RY_FIND and RY_FIND_NEXT functions.

4. Close the NetROCKEY4 dongle (RY_CLOSE)

Objective : To close a NetROCKEY4 service and logout.

Input parameters:

function = RY_CLOSE
*handle = The NetROCKEY4’s handle.

Return value:

A return value = “ 0” indicates that the function worked correctly. Any other return value indicates an error.



5. Read from a NetROCKEY4 dongle (RY_READ)

Objective : To read the contents of the User Data Zone (UDZ).

Input parameters:

function = RY_READ
*handle = NetROCKEY4's handle
*p1 = offset of UDZ
*p2 = length (unit is byte)
buffer= address of buffer

Return value:

A return value = “ 0” indicates that the function worked correctly. Any other return value indicates an error. A successful operation will result in the contents of the UDZ written to the memory buffer.

6. Write to a NetROCKEY4 dongle (RY_WRITE)

Objective : To write data to the User Data Zone. (UDZ)

Input parameters:

function = RY_WRITE
*handle = NetROCKEY4's handle
*p1 = offset of UDZ
*p2 = length (unit is byte)
buf = address of buffer

Return value:

A return value = “ 0” indicates that the function worked correctly. Any other return value indicates an error.

7. Generate a Random Number (RY_RANDOM)

Objective : To get a random number.

Input parameters:

function = RY_RANDOM
*handle = NetROCKEY4's handle

Return value:

A return value = “ 0” indicates that the function worked correctly. Any other return value indicates an error. A successful operation will result in the *p1 address populated with the random number.

8. Generate Seed Code Return Values (RY_SEED)



Objective: To get return codes from the input of a seed code.

Input parameters:

function = RY_SEED
*handle = NetROCKEY4's handle
*lp2 = Seed Code (32-bit)

Return value:

A return value = “ 0” indicates that the function worked correctly. Any other return value indicates an error. A successful operation will result in the following addresses populated with seed code return values:

*p1 = Return Code 1
*p2 = Return Code 2
*p3 = Return Code 3
*p4 = Return Code 4

9. Write the User ID (RY_WRITE_USERID)

Note: This function is not supported by NetROCKEY4. You may use the Editor program of standalone ROCKEY4 to write the user ID to NetROCKEY4.

10. Read User ID (RY_READ_USERID)

Objective: To read the user defined “ User ID” from the User ID Zone. (UIZ)

Input parameters:

Function = RY_READ_USERID
*handle = NetROCKEY4's handle

Return value:

A return value = “ 0” indicates that the function worked correctly. Any other return value indicates an error. A successful operation will result in the *lp1 address populated with the data stored in the UIZ.

11. Set a NetROCKEY4 Module (RY_SET_MOUDLE)

Note: This function is not supported by NetROCKEY4. You may use the Editor program of standalone ROCKEY4 to set the module of NetROCKEY4.

12. Check a NetROCKEY4 Module (RY_CHECK_MOUDLE)

Objective: To read the attributes of a specific NetROCKEY4 module.

**Input parameters:**

function = RY_CHECK_MOUDLE
*handle = NetROCKEY4's handle
*p1 = Module Number

Return value:

A return value = “ 0” indicates that the function worked correctly. Any other return value indicates an error. A successful operation will result in “*p2” populated with the value from the Zero Value attribute (1 = module is not zero), and “*p3” populated with the value from the Decrement attribute (1 = module can be decreased).

13. Write Arithmetic (RY_WRITE_ARITHMETIC)

Note: This function is not supported by NetROCKEY4. You may use the Editor program of standalone ROCKEY4 to write the algorithms to NetROCKEY4.

14. Calculate 1 (RY_CALCULATE1)

Objective : To return the results of calculation 1 performed in NetROCKEY4. (See Chapter 8)

Input parameters:

function = RY_CALCULATE1
*handle = NetROCKEY4's handle
*lp1 = Start point of calculation from the UAZ
*lp2 = Module number
*p1 = Input value 1
*p2 = Input value 2
*p3 = Input value 3
*p4 = Input value 4

Return value:

A return value = “ 0” indicates that the function worked correctly. Any other return value indicates an error. A successful operation will result in the following addresses populated with the results of the instruction:

*p1 = Return value 1
*p2 = Return value 2
*p3 = Return value 3
*p4 = Return value 4

15. Calculate 2 (RY_CALCULATE2)

Objective : To return the results of calculation 2 performed in NetROCKEY4. (See Chapter 8)

**Input parameters:**

function = RY_CALCULATE2
*handle = NetROCKEY4's handle
*lp1 = Start point of calculation from the UAZ
*lp2 = Seed Code (32-bit)
*p1 = Input value 1
*p2 = Input value 2
*p3 = Input value 3
*p4 = Input value 4

Return value:

A return value = “ 0” indicates that the function worked correctly. Any other return value indicates an error. A successful operation will result in the addresses p1, p2, p3 and p4 populated with the results of the instruction.

*p1 = Return value 1
*p2 = Return value 2
*p3 = Return value 3
*p4 = Return value 4

16. Calculate 3 (RY_CALCULATE3)

Objective : To return results of calculation 3 performed in NetROCKEY4. (See Chapter 8)

Input parameters:

function = RY_CALCULATE3
*handle = NetROCKEY4's handle
*lp1 = Start point of calculation from UAZ
*lp2 = Module number
*p1 = Input value 1
*p2 = Input value 2
*p3 = Input value 3
*p4 = Input value 4

Return value:

A return value = “ 0” indicates that the function worked correctly. Any other return value indicates an error. A successful operation will result in the addresses p1, p2, p3 and p4 populated with the results of the instruction.

*p1 = Return value 1
*p2 = Return value 2
*p3 = Return value 3
*p4 = Return value 4

17. Decrease Module Unit (RY_DECREASE)



Objective : To decrease the value in a specified NetROCKEY4 module by “ I”.

Input parameters:

function = RY_DECREASE
*handle = NetROCKEY4's handle
*p1 = Module Number

Return value:

A return value = “ 0” indicates that the function worked correctly. Any other return value indicates an error. A successful operation will reduce the value stored in module *p1 by “ I”.

9.10 NetROCKEY4 Return Codes

There are two kinds of return codes: Normal and Extended. Normal return codes are the return values of the NetROCKEY4 API. Extended return codes are values returned from the NrGetLastError function. Extended return codes are related to network issues.

9.10.1 Normal Return Codes

Normal Return Codes		Return Code Description
ERR_SUCCESS	0	Success
ERR_NO_PARALLEL_PORT	1	No parallel port on the computer
ERR_NO_DRIVER	2	No driver installed
ERR_NO_ROCKEY	3	No ROCKEY4 dongle
ERR_INVALID_PASSWORD	4	ROCKEY4 dongle found, but basic password is incorrect
ERR_INVALID_PASSWORD_OR_ID	5	Wrong password or ROCKEY4 HID
ERR_SETID	6	Set ROCKEY4 HID wrong
ERR_INVALID_ADDR_OR_SIZE	7	Read/Write address is wrong
ERR_UNKNOWN_COMMAND	8	No such command
ERR_NOTBELEVEL3	9	Inside error
ERR_READ	10	Read error
ERR_WRITE	11	Write error
ERR_RANDOM	12	Random error
ERR_SEED	13	Seed Code error
ERR_CALCULATE	14	Calculate error
ERR_NO_OPEN	15	Ry_Open must precede this operation
ERR_OPEN_OVERFLOW	16	Too many open dongles (>16)
ERR_NOMORE	17	No more dongle
ERR_NEED_FIND	18	No Find before FindNext
ERR_DECREASE	19	Decrease error

Normal Return Codes		Return Code Description
ERR_AR_BADCOMMAND	20	Arithmetic instruction error
ERR_AR_UNKNOWN_OPCODE	21	Arithmetic operator error
ERR_AR_WRONGBEGIN	22	A constant. cannot be in the first instruction
ERR_AR_WRONG_END	23	A constant. cannot be in the last instruction
ERR_AR_VALUEOVERFLOW	24	Const number > 63
ERR_NET_LOGINAGAIN	1001	A module can only be opened once by the same process.
ERR_NET_NETERROR	1002	Network error.
ERR_NET_LOGIN	1003	Too many users are logged on.
ERR_NET_INVALIDHANDLE	1004	Invalid handle, this handle might have been closed.
ERR_NET_BADHARDWARE	1005	Defective hardware
ERR_NET_REFUSE	1006	Client dll modified, service refused request.
ERR_NET_BADSERVER	1007	Nrsrv.exe modified, service is invalid.

Below are network error codes. NrGetLastError function can return extended return codes.

ERR_INIT SOCK	2001	Error when initializing.
ERR_NOSUCHPROTO	2002	No such protocol.
ERR_UDPSOCKCREATE	2003	UDP error when creating socket.
ERR_UDPSETBROADCAST	2004	UDP error when setting broadcast.
ERR_UDPBINDFAILED	2005	UDP error when binding.
ERR_SVRCALLBACKNULL	2006	Server call back null.
ERR_TCPSOCKCREATE	2007	TCP error when creating socket.
ERR_TCPBINDFAILED	2008	TCP error when binding.
ERR_TCPLISTENFAILED	2009	TCP error when listening.
ERR_NOSUCHSEARCH	2010	No such search mode.
ERR_UDPSEND	2012	UDP error when sending.
ERR_UDPTIMEOUT	2013	UDP timeout error when waiting.
ERR_UDP_RECV	2014	UDP error when receiving.
ERR_TCPCONNECT	2015	TCP error when connecting to server.
ERR_TCPSENDTIMEOUT	2016	TCP time out error when sending.
ERR_TCPSEND	2017	TCP error when sending.
ERR_TCPRECVTIMEOUT	2018	TCP time out error when receiving.
ERR_TCPRECV	2019	TCP error when receiving.
ERR_IPXSOCKCREATE	2020	IPX error when creating socket.
ERR_IPXSETBROADCAST	2021	IPX error when setting broadcast.
ERR_IPXSEND	2022	IPX error when sending data.
ERR_IPXRECV	2023	IPX error when receiving data.

Normal Return Codes		Return Code Description
ERR_IPXBIND	2024	IPX error when binding.
ERR_NBSRESET	2025	NetBIOS error when initializing.
ERR_NBSADDNAME	2026	NetBIOS error when adding name.
ERR_NBSSEND	2027	NetBIOS error when sending data.
ERR_NBSRECV	2028	NetBIOS error when receiving data.
ERR_UNKNOWN	0xffff	Unknown error.

9.10.2 NetROCKEY4 Extended Return Codes

Run the NrGetLastError function after receiving any network related return code. The result will be an error code constant that you can use to look up more detailed error information in the TCP/UDP or IPX specifications.

UDP/TCP and IPX Extended Return Codes: (Reference)

Definition Statement	Regular Berkeley Error Constants
#define WSABASEERR	10000
#define WSAEINTR	(WSABASEERR+4)
#define WSAEBADF	(WSABASEERR+9)
#define WSAEACCES	(WSABASEERR+13)
#define WSAEFAULT	(WSABASEERR+14)
#define WSAEINVAL	(WSABASEERR+22)
#define WSAEMFILE	(WSABASEERR+24)
#define WSAEWOULDBLOCK	(WSABASEERR+35)
#define WSAEINPROGRESS	(WSABASEERR+36)
#define WSAEALREADY	(WSABASEERR+37)
#define WSAENOTSOCK	(WSABASEERR+38)
#define WSAEDESTADDRREQ	(WSABASEERR+39)
#define WSAEMSGSIZE	(WSABASEERR+40)
#define WSAEPROTOTYPE	(WSABASEERR+41)
#define WSAENOPROTOOPT	(WSABASEERR+42)
#define WSAEPROTONOSUPPORT	(WSABASEERR+43)
#define WSAESOCKTNOSUPPORT	(WSABASEERR+44)
#define WSAEOPNOTSUPP	(WSABASEERR+45)
#define WSAEPFNOSUPPORT	(WSABASEERR+46)
#define WSAEAFNOSUPPORT	(WSABASEERR+47)
#define WSAEADDRINUSE	(WSABASEERR+48)
#define WSAEADDRNOTAVAIL	(WSABASEERR+49)
#define WSAENETDOWN	(WSABASEERR+50)
#define WSAENETUNREACH	(WSABASEERR+51)

Definition Statement	Regular Berkeley Error Constants
#define WSAENETRESET	(WSABASEERR+52)
#define WSAECONNABORTED	(WSABASEERR+53)
#define WSAECONNRESET	(WSABASEERR+54)
#define WSAENOBUFS	(WSABASEERR+55)
#define WSAEISCONN	(WSABASEERR+56)
#define WSAENOTCONN	(WSABASEERR+57)
#define WSAESHUTDOWN	(WSABASEERR+58)
#define WSAETOOMANYREFS	(WSABASEERR+59)
#define WSAETIMEDOUT	(WSABASEERR+60)
#define WSAECONNREFUSED	(WSABASEERR+61)
#define WSAELOOP	(WSABASEERR+62)
#define WSAENAMETOOLONG	(WSABASEERR+63)
#define WSAEHOSTDOWN	(WSABASEERR+64)
#define WSAEHOSTUNREACH	(WSABASEERR+65)
#define WSAENOTEMPTY	(WSABASEERR+66)
#define WSAEPROCLIM	(WSABASEERR+67)
#define WSAEUSERS	(WSABASEERR+68)
#define WSAEDQUOT	(WSABASEERR+69)
#define WSAESTALE	(WSABASEERR+70)
#define WSAEREMOTE	(WSABASEERR+71)
#define WSASYSNOTREADY	(WSABASEERR+91)
#define WSAVERNOTSUPPORTED	(WSABASEERR+92)
#define WSANOTINITIALISED	(WSABASEERR+93)
#define WSAEDISCON	(WSABASEERR+101)
#define WSAHOST_NOT_FOUND	(WSABASEERR+1001)
#define WSATRY AGAIN	(WSABASEERR+1002)
#define WSANO RECOVERY	(WSABASEERR+1003)
#define WSANO DATA	(WSABASEERR+1004)

NetBIOS Extended Return Codes: (Reference)

Return Code	Return Code Definition
0x00	Good return, also returned when ASYNCH request accepted
0x01	Illegal buffer length
0x03	Illegal command
0x05	Command time out
0x06	Message incomplete, issue another command
0x07	Illegal buffer address
0x08	Session number out of range
0x09	No resource available
0x0a	Session closed

Return Code	Return Code Definition
0x0b	Command cancelled
0x0d	Duplicate name
0x0e	Name table full
0x0f	No deletions, name has active sessions
0x11	Local session table full
0x12	Remote session table full
0x13	Illegal name number
0x14	No call name
0x15	Cannot put * in NCB_NAME
0x16	Name in use on remote adapter
0x17	Name deleted
0x18	Session ended abnormally
0x19	Name conflict detected
0x21	Interface busy, IRET before retrying
0x22	Too many commands outstanding, retry later
0x23	ncb_lana_num field invalid
0x24	Command completed while cancel occurring
0x26	Command not valid to cancel
0x30	Name defined by another local process
0x34	Environment undefined. RESET required
0x35	Required OS resources exhausted
0x36	Max number of applications exceeded
0x37	No saps available for NetBIOS
0x38	Requested resources are not available
0x39	Invalid ncb address or length > segment
0x3B	Invalid NCB DDID
0x3C	Lock of user area failed
0x3f	NETBIOS not loaded
0x40	System error
0xff	Synchronous command is not yet finished

9.11 NetROCKEY4 Application Example

This NetROCKEY4 program example is simplified and intended for educational purposes. Developers will have to determine methods suitable to their licensing policies and user requirements.

```
#include "stdafx.h"
#include "conio.h"
#include "stdlib.h"

#include "nrclient.h"                                //Header file of NrClient.dll
```

```
void ShowError(HRESULT err); //Function to print error code.

NETROCKEY      pfnNetRockey=NULL; //Pointer of function in NrClient.dll
NRGETLASTERROR pfnNrGetLastError=NULL; //Pointer of function in NrClient.dll
NRSETINIPNAME   pfnSetIniPathName=NULL; //Pointer of function in NrClient.dll

int main()
{
    //Load NrClient.dll to process memory space.
    HMODULE hDll=LoadLibrary("NrClient.dll");

    if(!hDll)
    {
        printf("can't find NrClient.dll, please copy and try again.\n");
        return 0;
    }

    //Locate entry address of function in NrClient.dll

    pfnNetRockey=(NETROCKEY)GetProcAddress(hDll,"NetRockey");
    pfnNrGetLastError=(NRGETLASTERROR)GetProcAddress(hDll,"NrGetLastError");
    pfnSetIniPathName=(NRSETINIPNAME)GetProcAddress(hDll,"SetIniPathName");

    if(!pfnNetRockey||!pfnNrGetLastError||!pfnSetIniPathName)
    {
        printf("This is a wrong NrClient.dll, please replace it.\n");
        FreeLibrary(hDll);
        return 0;
    }

    DWORD      hResult;           //errcode of Function.
    WORD       handle;            //record opened handle.
    WORD       p1, p2, p3, p4;    //WORD parameters.
    DWORD      lp1, lp2;          //DWORD parameters.
    BYTE       buffer[1024];       //Buffer

    DWORD      hID[256];          //record HardID of found Net-Rockey.
    DWORD      dwMaxRockey=0;     //total number of found Net-Rockey.

    //Find Net-Rockey, using DEMO password.
    //Net-Rockey can't be write in anything when p3=p4=0;
    printf("***** Finding Net-Rockeys ... \n");
    p1 = 0xc44c;
    p2 = 0xc8f8;
    p3 = 0;
    p4 = 0;
    dwMaxRockey=0;
    hResult=pfnNetRockey(RY_FIND,&handle,&lp1,&lp2,&p1,&p2,&p3,&p4,buffer);
    if(ERR_SUCCESS==hResult)
```

```
{  
    //lp1      ---- HardID of Founded Net-Rockey.  
    //buffer   ---- computer name of Net-Rockey insert in.  
    printf("Found, (%d) HID:%lx @ %s\n",dwMaxRockey,lp1,buffer);  
    hID[dwMaxRockey]=lp1;  
    dwMaxRockey++;  
}  
else  
{  
    //Show Error and exit program.  
    ShowError(hResult);  
    FreeLibrary(hDll);  
    return 0;  
}  
  
//find other Net-Rockeys.  
while(1)  
{  
    hResult=pfnNetRockey(RY_FIND_NEXT,&handle,&lp1,&lp2,&p1,&p2,&p3,&p4,buffer);  
    if(ERR_SUCCESS==hResult)  
    {  
        printf("Found, (%d) HID:%lx @ %s\n",dwMaxRockey,lp1,buffer);  
        hID[dwMaxRockey]=lp1;  
        dwMaxRockey++;  
    }  
    else break;  
}  
  
printf("\n***** Select NetROCKEY and module to open,(exp.Input 0 0)\n");  
int iRockey,iModule;  
scanf("%d %d",&iRockey,&iModule);  
  
//Open a module  
printf("\n***** Opening module %d of %x ... \n",iModule,hID[iRockey]);  
lp1=hID[iRockey];  
lp2=(DWORD)iModule;  
hResult=pfnNetRockey(RY_OPEN,&handle,&lp1,&lp2,&p1,&p2,&p3,&p4,buffer);  
if(ERR_SUCCESS==hResult)  
{  
    printf("Opened,Handle:%d\n",handle);  
}  
else  
{  
    ShowError(hResult);  
    FreeLibrary(hDll);  
    exit(0);  
}  
  
printf("\n***** Press any key to begin testing...\n");
```

```

getch();

//Read User Memory.
printf("\n***** Reading user memory ...\\n");

BYTE rbuffer[30];
p1=0; p2=24;
hResult=pfnNetRockey(RY_READ,&handle,&lp1,&lp2,&p1,&p2,&p3,&p4,rbuffer);
if(ERR_SUCCESS==hResult)
{
    rbuffer[24]=0;
    printf("Succeed, %s\\n",rbuffer);
}
else

{
    ShowError(hResult); goto ENDPROGRAM;
}

//Random number test.
printf("\n***** Testing Random function...\"");
int i;
for(i=0; i<5;i++)
{
    hResult=pfnNetRockey(RY_RANDOM,&handle,&lp1,&lp2,&p1,&p2,&p3,&p4,rbuffer);
    if(ERR_SUCCESS==hResult) printf("%8x\\n",p1);
    else
    {
        ShowError(hResult); goto ENDPROGRAM;
    }
}
printf("\\n");

//Seed test.
lp2=0xF025;
printf("\n***** Seed test,seed is %x\\n",lp2);
hResult=pfnNetRockey(RY_SEED,&handle,&lp1,&lp2,&p1,&p2,&p3,&p4,rbuffer);
if(ERR_SUCCESS==hResult) printf("%8x%8x%8x%8x\\n",p1,p2,p3,p4);
else
{
    ShowError(hResult); goto ENDPROGRAM;
}

//Read User ID
lp1=0;
printf("\n***** Reading UserID...\\n");
hResult=pfnNetRockey(RY_READ_USERID,&handle,&lp1,&lp2,&p1,&p2,&p3,&p4,rbuffer);
if(ERR_SUCCESS==hResult) printf("Succeed,ID = %08lx\\n",lp1);
else

```

```

{
    ShowError(hResult); goto ENDPROGRAM;
}

//Check the property of module 0
p1=0;
printf("\n***** Checking module 0...\n");
hResult=pfnNetRockey(RY_CHECK_MOUDLE,&handle,&lp1,&lp2,&p1,&p2,&p3,&p4,rbuffer);
if(ERR_SUCCESS==hResult)
{
    printf("Succeed,validity:%d,can decrease:%d\n",p2,p3);

    /*
    if(p3)
    {
        //you can decrease this module by using below code
        p1=0;
        printf("\n***** Decrease module 0...\n");

        hResult=pfnNetRockey(RY_DECREASE,&handle,&lp1,&lp2,&p1,&p2,&p3,&p4,rbuffer);
        if(ERR_SUCCESS==hResult) printf("Succeed.\n");
    }
    */
}
else

{
    ShowError(hResult); goto ENDPROGRAM;
}

//Calculate 1
lp1=0;
lp2=0;
p1=1;p2=2;p3=3;p4=4;
printf("\n***** Calculate 1 test,Startposition %d\n",lp1);
hResult=pfnNetRockey(RY_CALCULATE1,&handle,&lp1,&lp2,&p1,&p2,&p3,&p4,buffer);
if(ERR_SUCCESS==hResult)
    printf("Succeed,p1=%d,p2=%d,p3=%d,p4=%d\n",p1,p2,p3,p4);
else
{
    ShowError(hResult); goto ENDPROGRAM;
}

//Calculate 2
lp1=0;
lp2=0;
p1=1;p2=2;p3=3;p4=4;
printf("\n***** Calculate 2 test,Startposition %d\n",lp1);

```

```

hResult=pfnNetRockey(RY_CALCULATE2,&handle,&lp1,&lp2,&p1,&p2,&p3,&p4,buffer);
if(ERR_SUCCESS==hResult)
    printf("Succeed,p1=%d,p2=%d,p3=%d,p4=%d\n",p1,p2,p3,p4);

else
{
    ShowError(hResult); goto ENDPROGRAM;
}

//Calculate 3
lp1=0;
lp2=0;
p1=1;p2=2;p3=3;p4=4;
printf("\n***** Calculate 3 test,Startposition %d\n",lp1);
hResult=pfnNetRockey(RY_CALCULATE1,&handle,&lp1,&lp2,&p1,&p2,&p3,&p4,buffer);
if(ERR_SUCCESS==hResult)
    printf("Succeed,p1=%d,p2=%d,p3=%d,p4=%d\n",p1,p2,p3,p4);
else
{
    ShowError(hResult); goto ENDPROGRAM;
}

ENDPROGRAM:

//close NetROCKEY.
pfnNetRockey(RY_CLOSE,&handle,&lp1,&lp2,&p1,&p2,&p3,&p4,buffer);
printf("Handle %x Closed.\n",handle);

FreeLibrary(hDll);
return 0;
}

void ShowError(HRESULT hResult)
{
    char temp[80];

    if(hResult<2000)sprintf(temp,"Normal error,code %d\n",hResult);
    else
    {
        sprintf(temp,"Net error,code %d,Extended errcode %d\n",hResult,
               pfnNrGetLastError());
    }
    printf(temp);
}

```



9.12 Quick Test

1. Determine the network protocols that you will need (IPX, TCP/UDP and/or NetBIOS). Verify that you have all the contents of the developer's kit: NetROCKEY4 dongle, CD-ROM and developer's Guide. Here we take TCP/UDP protocol as the example.
2. Test physical network connectivity by running a "ping" test between the client and server machines. Verify the correct protocols are installed on the client and service program PCs.
3. Run *setup.exe* on the server and client machines. Install all of the components on the server. Install only the NetROCKEY4 component on the client machine(s). The client does not need the NetROCKEY4 drivers.
4. Insert the NetROCKEY4 dongle into the LPT or USB port of the server. Use the Editor program to write the maximum number of simultaneous users to a NetROCKEY4 module. For example, write "2" to module number 0 to allow a maximum of two users to access the software at any one time.
5. Start the NetROCKEY4 service by running the *nrsvr.exe* program. The service program will use the default configuration file (*svrcfg.ini*). After you change the configuration file, you have to exit and restart the service program. Use the NrMon to view the status of the service program.
6. Run the *nrtest.exe* program from a client computer. Attempt to log into the module where you wrote the maximum number of users. In the example given above, you should only be able to log into module number "0" two times. Use the NrMon to view the status of the service and client programs. By default we automatically search in UDP protocol, so it is not necessary to specify the address of server in client configuration file.

Chapter 10

Frequently Asked Questions

This chapter will help you to resolve some questions you or your customers may run into when use our ROCKEY. We list the questions and the answers here for your reference.

10.1 Trouble Shooting Suggestions

- ◊ Use the Self Test program in the ROCKEY4 Editor to test the dongle. Refer to the Editor chapter in this document.
- ◊ Be sure you are using the latest version of the ROCKEY4 driver. Download the driver from our website at: <http://www.FTsafe.com>
- ◊ Please refer to our website; we update its content frequently.
- ◊ Test the dongle with another computer.
- ◊ If there is a printer connected to the LPT dongle, try removing the printer and testing the dongle.
- ◊ If there is a secondary parallel or USB port, try testing the dongle on one of these ports.
- ◊ Verify that your computer is not infected with a virus.

10.2 FAQs

1. What is the software developer's kit or evaluation kit?

The Software Developer's Kit (SDK) is a set of evaluation package we send to developers for the purpose of evaluation. It includes one Developer's Guide, one CD-ROM, one USB port ROCKEY dongle and/or one parallel port ROCKEY dongle. The only difference between the SDK dongle and the commercial dongle is that the passwords to the SDK dongle are well known. After evaluating and accepting the ROCKEY SDK, the developer may order dongles with secure passwords.

2. What is the purpose of the ROCKEY4 purchase code?

The purchase code is for order management. You should use it if you need to reorder with the same configuration as your previous order.

3. Can another customer purchase the same dongles as mine?

Feitian's manufacturing process guarantees that each ROCKEY4 dongle is unique. The Hardware Identifier (HID) is a globally unique serial number that is burned into each dongle and cannot be

changed by anyone, including the manufacturer. Passwords are specific to a customer and cannot be duplicated for another customer. It is our mission to protect the assets of our customers and we will even enter into separate security agreements with our customers if that is required.

4. Is the ROCKEY4 password secure enough?

Yes. It is very secure. The ROCKEY4 password is actually a set of four individual passwords that authorize two security levels. The two Basic passwords authorize read only access to the dongle. Read and Write access is authorized with only the combination of the Basic and Advanced password sets. The Advanced password set is meant for the developer and should not appear in the software that goes to the end user. Also, if the incorrect Advanced level passwords have been written four consecutive times, the ROCKEY4 dongle will lock for 2 seconds. No operation can be accepted during this “locked” period. This feature is intended to protect ROCKEY4 from hackers who attempt programmatic methods of cracking the dongle.

5. What are “same-number” ROCKEY4s?

“Same-number” ROCKEY4 dongles are a set of dongles that have the same passwords. There will be a purchase code that corresponds to the “same-number” dongles. It is necessary for most of our customers that all of their dongles have the same password so that they only need compile their programs once before distribution.

6. What if I forgot my password(s)?

The simplest way is to order dongles with new passwords. Another way is to provide proof that you are the legitimate holder of the purchase code. Only the legitimate purchase code holder may obtain the passwords from Feitian.

7. Will the parallel port ROCKEY4 dongle conflict with my printer?

ROCKEY4 operates the parallel port through the device driver, and uses error correction technology at the communication protocol level. Therefore, it is compatible with most printers. However, there are a few models of printers that receive a large amount of current when powered off. This can cause a voltage shortfall to the dongle and result in a communication problem with the dongle driver. This problem may be easily solved by turning on the printer, or removing the printer from the dongle.

8. How to improve the compatibility of ROCKEY?

The old version of the ROCKEY4 driver only supported normal printing mode. The newer version support normal, EPP and ECP modes. Be sure you are using the latest version of our driver. You may download the driver from our Website, www.FTsafe.com Another solution to this problem would be to simply switch to the USB version of the ROCKEY4 dongle. There is no programming required to switch to USB.



The compatibility of the LPT dongle and printer may be enhanced by setting the port in CMOS setup to EPP or ECP. An example using the AWARD system BIOS is given below:

Reboot the computer, press DEL key to enter CMOS Setup mode.

Select the menu option "INTEGRATED PERIPHERALS".

Then use either PAGEUP or PAGEDOWN key to change the "Parallel Port Mode" option.

9. There is a kind of device in the marketplace that emulates a dongle. What is the principle of such device? Does ROCKEY4 have any countermeasures?

There are emulation programs that record all of the port accesses between the dongle and the driver. Then the emulation software attempts to “playback” the accesses after the dongle is removed. ROCKEY4 has a built in countermeasure for this sort of attack. Low layer communications between the dongle, driver and application are encoded. There is also “random noise” introduced to the protocol communications so that any information recorded by the emulation program will be invalid from session to session. In addition, emulation software of this type will not work with USB attached devices. You may implement your own countermeasure. The judicious use of seed code protection methods will make any such attack very difficult to implement successfully.

10. Can the data sharing device share ROCKEY?

It is quite easy to prevent data sharing devices. Generate a random number when the program starts and write this number to a specified address in ROCKEY. During run time the program will check the number in the specified address to see if it is the number it has written. If another computer also runs this program during this period, and uses the dongle, it will overwrite the random number and lock the other computer out of the application.

11. Will the performance of my application be adversely affected by a complex ROCKEY4 algorithm?

No. According to the results of our tests, the performance difference between the simplest algorithm and the most complex was within a dozen milliseconds. You will notice no difference as long as you do not call the algorithm too frequently.

12. What is a USB port? What are its advantages?

USB stands for Universal Serial Bus. It is a new interface standard. Please visit www.usb.org for details. Its advantages include plug and play, hot swap support and a high transfer speed. It can support up to 127 USB devices connected by extension. USB eliminates conflict problems with printers and other peripheral devices.

13. I used the parallel port ROCKEY4 to protect my application. Now I would like to change to the USB model. Do I need to modify my application?

No. They are fully compatible, other than the requirement that you install the USB driver. The USB *ROCKEY4 Developer's Guide*



device will prompt you, automatically, to install the driver.

14. Why does my screen display an unknown device after I plug in the USB ROCKEY4?

It may be caused by interference or a bad contact. Try to re-plug the device.

15. My computer has a USB port and Windows 98. Why is the USB device not listed in the device manager?

It is possible that the USB support option inside the BIOS has been disabled.

16. How to update ROCKEY software?

You may download the latest ROCKEY software from our website <http://www.FTsafe.com> free of charge.

17. After the program written in FoxPro or VB is encrypted with API call, RYDLL32.DLL cannot be found. Why?

Though *RYDLL32.DLL* is under the current directory, FoxPro and VB programs search DLL only under the system directory. Copy *RYDLL32.DLL* to the system directory.

18. The system can not find parallel port ROCKEY, why?

A: The printer attaches to the ROCKEY is powered off.

If a printer or any other peripheral device is attached to the parallel ROCKEY you should power on the device. The ROCKEY dongle requires a voltage about 2.2 volts and this power is supplied by the main board through the parallel port. The system may not be able to find the ROCKEY dongle if an attached device is drawing too much voltage.

B: ROCKEY does not support the parallel mode.

The diversity of computers leads to many different parallel modes. ROCKEY dongles support most modes but for some special modes it is possible that the system cannot find the dongle or there is a conflict. What you need to do is just to change the parallel mode, set the port in CMOS setup to ECP. If the operating system is Windows NT/2000 you may have to reinstall the system after you change the parallel modes.

C: The drivers are not properly installed.

ROCKEY requires the installation of drivers, so the settings of drivers may affect the operation of ROCKEY. There are 4 options when we install the drivers: "Install ROCKEY parallel driver", "Install ROCKEY USB driver", "Detect-print-busy mode" and "Not-detect-print-busy mode". We suggest users choose "Install ROCKEY parallel driver", "Install ROCKEY USB driver", and "Detect-print-busy mode". In Windows NT/2000 only the administrator has the privilege to install drivers.



D: ROCKEY is not compatible with other devices.

The parallel port was not designed to connect many devices at the same time, and in theory only one device can be attached to the parallel port at one time. With the users in mind, our ROCKEY was designed to allow the standard devices, such as printer and scanner, to attach to it, but for some special devices we can not guarantee that our ROCKEY is compatible with them. Plug out the device to see if it is compatibility problem.

19. The system can not find USB ROCKEY?

Maybe the drivers are not properly installed, please refer to solution C to Question 18.

The LED indicator lamp may help you to find the reason. If the indicator is bright all the time, it indicates the dongle is working properly. If it winks once every second, the drivers are not installed or something wrong with the USB port. Otherwise the dongle is defective. If the indicator is not bright and the system prompts “Found USB Device”, it indicates the dongle is defective. Otherwise there is a bad connection or something is wrong with the USB port. To check the USB port: when you first attach ROCKEY to the computer the system should prompt “Found USB Device” and install the driver. If this does not occur the USB port must get some problem. This may be caused by the incorrect installation of the driver for USB controller or the USB function is disabled in COMS SETUP or hardware failure of the USB port of main board. You may test the USB port with another USB device such as a USB mouse.

20. How to prevent LPT ROCKEY from accidental damages?

Because the LPT ROCKEY works between the computer and peripheral device, it is affected by both of them. Most ROCKEY dongles are damaged by over voltage. Please be sure the PC is grounded to avoid this problem – the ground wire should connect the computer case to a ground. If it is not possible to ground the computer, be sure the power is off when connecting the dongle to the PC.

21. Why I can not find a specific dongle when I attach them to the same parallel port?

When users have several applications protected with different ROCKEY, they may cascade these dongles together. In most cases there is no problem for them to work together, but sometimes when you try to use one application you are told that “Can not find ROCKEY”.

When the system sends its request to ROCKEY, a value will be returned to specify the next operation. If several dongles are cascaded, every dongle will receive the request and respond to it. Simultaneous return signals can cause interference that prevents the system from recognizing the dongle.

22. After a period of time the dongle does not work. Why?

If the problem is resolved after a restart you probably have not set a value for p3 and p4. After a period of time ROCKEY return the message, “Open too many dongles” and it has to be opened again. Set a fixed value, such as “0” to p3 and p4. If restarting your computer does not work, maybe the

dongle is defective, or your computer is infected with virus, or the parallel port settings have been changed, or some other device is attached to ROCKEY, or the system has been reinstalled.

23. Why does the client or service program run ignoring information in the configuration files?

Both the client and service programs will use default settings if they do not find the configuration files. The configuration file path for the client program may be specified with the SetIniPathName API. By default the client and service programs will look to the current directory for their configuration files. Verify that the configuration files are either in the current directory, or that the path is correctly specified with the SetIniPathName API.

24. I started the service program but the Monitor can't find it.

Please check the following:

Verify that the machine where the Monitor program is running has the correct protocol installed. For example, you must have the IPX protocol installed on the Monitor program machine if you want to monitor IPX services.

Verify that the port number used by the service and client programs is the same.

Verify that the connection request is not being thwarted by a firewall.

25. I started the service program and the Monitor can find it, but my client programs cannot connect to the service.

This can be caused by:

The NetROCKEY4 dongle is not inserted into the server.

The driver for the dongle is not installed correctly.

Also check the points listed in item number 24.

26. My customer has two (or more) applications using NetROCKEY4. How can I avoid conflict?

There are two solutions: (1) Install the service programs on separate computers. This is the easiest way to deal with the problem. (2) If the service programs need to run on the same computer, consolidate the clients to a single service program. They will not conflict with each other because they have different passwords.

Appendix A

Content Structure of CD-ROM

<Api16>	ROCKEY API in Windows 3.1
<Samples>	API call samples for various languages
<Bc502>	Borland C++ 5.02
<Delphi10>	Borland Delphi 1.0
<TB3>	Tool Book 3.0
<Vb3>	Visual Basic 3.0
<Vb4>	Visual Basic 4.0
<Vc152>	Visual C/C++ 1.52
<Api32>	ROCKEY API in Windows 95/98/NT/2000/XP/2003
<ActiveX>	ActiveX control
<Samples>	API call samples for various languages
<Delphi4>	Delphi 4.0
<JScript>	Java Script
<Vb6>	Visual Basic 6.0
<VbScript>	VB Script
<Word>	Microsoft Word
<Bc>	Borland C/C++ API
<BC502>	Borland C/C++ 5.02 API
<Cbuilder>	C++ Builder API
<CB30>	C++ Builder 3.0 API
<CB40>	C++ Builder 4.0 API
<CB50>	C++ Builder 5.0 API
<CB60>	C++ Builder 6.0 API
<Delphi>	Delphi API
<Delphi2>	Delphi 2.0 API
<Delphi3>	Delphi 3.0 API
<Delphi4>	Delphi 4.0 API
<Delphi5>	Delphi 5.0 API
<Delphi6>	Delphi 6.0 API
<Delphi7>	Delphi 7.0 API
<Dll>	32-bit DLL API
<Samples>	API call samples for various languages
<Access2k>	Microsoft Access 2000
<FoxPro>	Microsoft FoxPro 6.0
<PBuilder>	Power Builder
<Vb6>	Visual Basic 6.0
<Vb6str>	Character string parameter(for VBA)

<Vc6>	Visual C/C++ 6.0
<VcwMFC>	Visual C/C++ 6.0 MFC
<FPS>	Microsoft Fortran Power Station API
<Java>	Java API
<VC>	Visual C/C++ API
<Samples>	API call samples
<Arx>	AutoCAD Arx
<Vc6>	Visual C/C++ 6.0
<Vs.net>	Vs.net
<Apidos>	ROCKEY API in DOS
<Bc31>	Borland C/C++ 3.1 API
<Bc45>	Borland C/C++ 4.5 API
<DPMI16>	Borland C/C++ DPMI 16-bit API
<DPMI32>	Borland C/C++ DPMI 32-bit API
<FoxPro>	FoxPro 2.5/2.6 API
<Msc7>	Microsoft C/C++ 7.0 API
<QB45>	QBasic 4.5 API
<Tc20>	Turbo C 2.0 API
<WatcomC>	Watcom C 10.0 API
<Novell>	Novell 5.0 NLM API
<Driver>	ROCKEY driver installation package
<Instdll>	ROCKEY driver installation package DLL version
<Delphi>	InstDll call sample in Delphi 5.0
<IS>	InstDll call sample in Install Shield 5.0
<VB>	InstDll call sample in Visual Basic 6.0
<VC>	InstDll call sample in Visual C/C++ 6.0
<Help>	ROCKEY help document
<Beginer>	Programing samples for beginner
<Linux>	ROCKEY driver and samples in Linux
<FreeBSD>	ROCKEY driver and samples in FreeBSD
<MacOS>	ROCKEY driver and samples in MAC OS
<Net>	NetROCKEY developing package
<Client>	NetROCKEY DLL for client
<Samples>	NetROCKEY programming samples
<Server>	NetROCKEY service program
<Tools>	NetROCKEY developing tools
<Utility>	ROCKEY utility
<Editor>	ROCKEY Editor
<Envelop>	ROCKEY standalone Envelop

Appendix B

ROCKEY4 Evaluation Form

After years of research and development, Feitian Technologies Co., Ltd. has developed the most advanced software protection products available anywhere. We offer an evaluation package for software developers that require software protection against piracy. Now, please try using ROCKEY to protect your software. Fill out the form below to make a performance comparison between the ROCKEY4 and other products--and find out the difference that ROCKEY makes.

Items	ROCKEY4	Competing product		
		Yes	No	?
Support USB port	✓			
Compact design, the parallel port ROCKEY4 is only 39mm long	✓			
Built-in 8-bit CPU, 8 pins for parallel port ROCKEY4	✓			
Low voltage design with operating voltage down to 2.2V	✓			
Passwords and ID numbers are burned to CPU, even the manufacturer cannot change them	✓			
Provide Read/Write memory	✓			
Unique hardware ID for each ROCKEY4	✓			
Smoothly cascaded with other ROCKEY and other dongles	✓			
Same-number ROCKEY4s can also be cascaded	✓			
Universal, be able to work with printer connected	✓			
Printing and ROCKEY4 operations do not interfere with each other while working simultaneously	✓			
Support IEEE1284 standard	✓			
Support envelop encryption (direct encryption on .exe and .dll files), no software source codes needed	✓			
Be able to prevent tracing and cracking	✓			
User-defined hardware algorithms	✓			
Encrypted software can run under Windows 95/97/98/2000/NT/XP/2003	✓			
Two-level passwords control, advanced passwords do not appear in the end user software	✓			
Large capacity CPU program memory	✓			

ROCKEY

Provide user storage space, user memory may be extended up to 512 bytes	✓			
Built-in time gate to prevent software tracing	✓			
API ↔ driver ↔ port encryption communication protocol, preventing from dongle emulation program	✓			
Be able to encrypt multiple software and modules	✓			

Appendix C

ROCKEY4 Feedback Form

Dear Customer:

In order to ensure product quality and further improve our service, Feitian herein requires all customers fill in this form when they encounter problems and require Feitian to replace the defective ROCKEY4. Please mail this form along with the defective ROCKEY4 back to Feitian. Please use the **ROCKEDIT** under the Utility directory to test your ROCKEY4. If the ROCKEY4 passes your test, but there are some other problems such as: ROCKEY does not work well when connected with a printer, please describe in detail about the printer mode settings, printer model, and the operation results, etc. And please describe other problems in the same way in the form below.

Your Company Name: _____ TEL: _____

Contact Person: _____ Test by: _____

You returned _____ pcs of ROCKEY4(s) to Feitian Technologies Co., Ltd. on __ (M) __ (D) __ (Y).

Model: ROCKEY4 - LPT, ROCKEY4 – USB, ROCKEY4 – LPT+,
 ROCKEY4 - USB +, NetROCKEY4 - LPT, NetROCKEY4 - USB or _____

Purchase code: _____

Test result of your ROCKEY4 with ROCKEDIT?

Normal Abnormal

Normal, but _____

Detailed description of problems:

Please keep and make a copy of this form for future reference, thank you for your cooperation.