DS1954B
Crypto iButton™

FIPS 140-1 Non-Proprietary
Cryptographic Module Security Policy

Level 3 Validation

August 16, 1999

Table of Contents

1 Introduction

1.1 Purpose

This is a non-proprietary Cryptographic Module Security Policy for the Dallas Semiconductor DS1954B Crypto iButton ™ . This policy was prepared as part of FIPS 140-1 certification of the Crypto iButton. FIPS 140-1 (Federal Information Processing Standards Publication 140-1 -- *Security Requirements for Cryptographic Modules*) gives U.S. Government requirements for cryptographic modules, and defines the Security Policy as:

> "A precise specification of the security rules under which the cryptographic module must operate, including rules derived from the security requirements of this standard, and the additional security rules imposed by the manufacturer."

The DS1954B provides extraordinary security, meeting all FIPS 140-1 level 3 requirements, and some level 4 requirements. This security policy describes how the Crypto iButton meets these requirements, and how it can be operated in a secure fashion.

1.2 For more information

This document describes the operations and capabilities of the DS1954B Crypto iButton™ in the technical terms of a FIPS 140-1 cryptographic module security policy.

> For more detailed information about the Crypto iButton, please visit the iButton web site at http://www.ibutton.com. The web site contains non-technical descriptions of Dallas iButton products, technical specifications, product offerings, iButton functionality, iButton developer information, and much more.
>
> Fore more information about the FIPS 140-1 standard and validation program please visit the NIST web site at http://csrc.nist.gov/cryptval/.
>
> For answers to technical or sales related questions please refer to the contacts listed on the iButton web site at http://www.ibutton.com, or the Dallas Semiconductor web site at http://www.dalsemi.com.

1.3 Terminology

In this document the Dallas Semiconductor DS1954B Crypto iButton™ is referred to as the DS1954B, Crypto iButton, cryptographic module, or module. The Crypto iButton is also referred to as simply "iButton", although this term also applies collectively to other iButtons such as the DS1990, DS1994, or DS1920, which cannot perform computations.

2 The DS1954B Crypto iButton™

The Crypto iButton provides hardware cryptographic services such as secure private key storage, a high-speed math accelerator for 1024-bit public key cryptography, and secure message digest (hashing). In FIPS 140-1 terminology the Crypto iButton is a “multi-chip standalone cryptographic module”; however, the Crypto iButton actually provides all its services using a single silicon chip packaged in a 16mm stainless steel case. Thus, the iButton can be worn by a person or attached to an object for up-to-date information at the point of use. The steel button is rugged enough to withstand harsh outdoor environments, and is durable enough for a person to wear everyday on a digital accessory like a ring, key fob, wallet, or badge.

Figure 1 – The DS1954B Crypto iButton™ is laser–engraved in steel and silicon

2.1 The iButton Cryptographic Module

The cryptographic boundary for the iButton is the surrounding steel shell. This surrounding shell is factory-lasered with the module's unique 64-bit registration number as shown in Figure 2. The figure shows a button with registration number "1A1D2516"$_{16}$, which is engraved on the encased silicon chip.

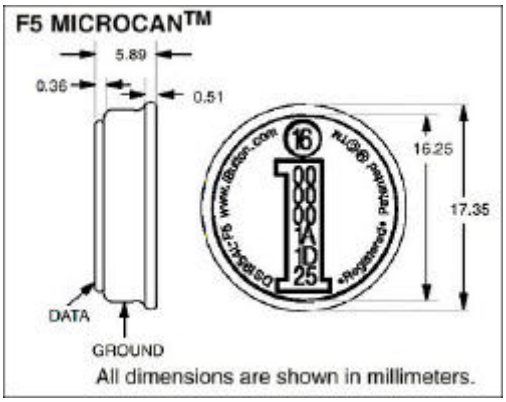

Figure 2 – Crypto iButton Case and Module Boundary

The ground side of the iButton may optionally be branded with logo facings. Registration numbers are also lasered into unalterable ROM on the iButtons, which can be read by any application communicating with an iButton. Strict factory controls ensure that registration numbers are globally unique, guaranteeing that no two iButtons ever share a registration number.

2.1.1 Module Interfaces

The button uses a single data contact on the front of the steel case to convey the module's five logical interfaces: data input, data output, control input, status output, and power. These interfaces are logically separated using the 1-Wire™ protocol, which regulates communications and separates reading, writing, and power applied to the module. The 1-Wire protocol utilizes a scratchpad buffer and features atomic, packetized transfers which assures error-free transmission, even with an intermittent connection, in addition to complete separation of input, processing, and output phases. Control commands and data must be input in error checked packets, and data and status are returned only after successful completion of processing.

2.1.2 Module Components

The active components of the iButton are shown below, and consist of a lithium cell (for backup power), an energy reservoir (to provide parasitic capacitance power), a quartz timing crystal (for a True Time Clock), and the single DS83C950 cryptographic chip.

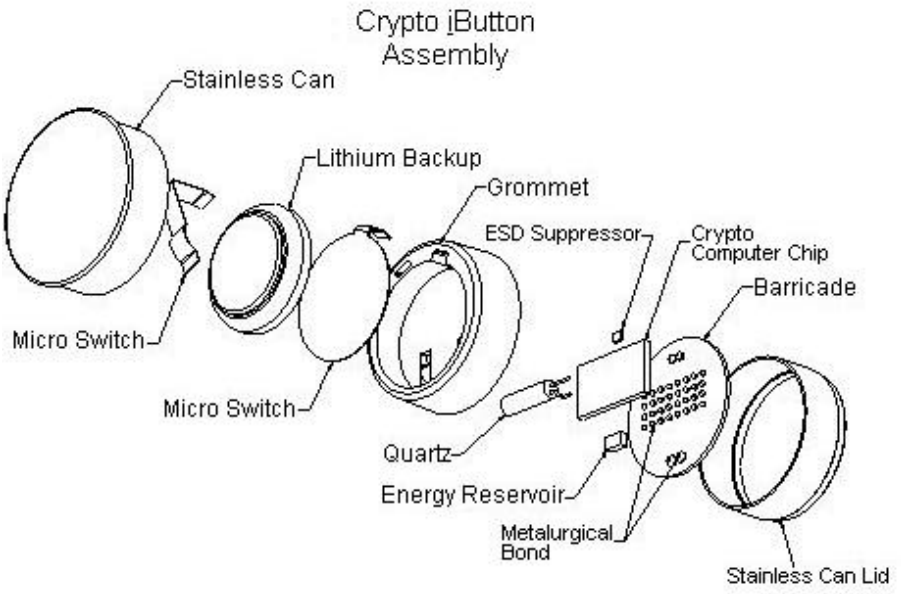

Figure 3 – Components of the DS1954B Crypto iButton™

2.2 Physical Security

The Crypto iButton boasts an incredible array of physical security safeguards packed into a small coin-sized device. Because the silicon chip is encased in stainless steel, the

iButton will stand up to the harsh conditions of daily wear, including dropping it, stepping on it, and inadvertently passing it through the washing machine and dryer.

Figure 3 – The Crypto iButton Mounted as a Signet Jewel of a Ring. It can be attached to any personal accessory

2.2.1 The Strength of Steel

The tough stainless steel case of the iButton also defines a contiguous perimeter and provides clear visual evidence of tampering. Tamper-signs include mangling and scratching of the data and ground plates and the smooth grommet separation. It is this outer case which satisfies FIPS 140-1 tamper evidence requirements for physical security. In addition, the case meets the FIPS 140-1 level 3 tamper response and level 4 Environmental Failure Protection (EFP) requirements.

2.2.2 Goes Down in a Blaze of Zeroization

If an iButton is pried open, a microswitch triggers an active zeroization of the chip's contents, destroying private keys and other sensitive information. The iButton constantly monitors the switch's contacts, and any separation of the cryptographic chip from the lithium cell switches the device to on-chip capacitor power to perform a complete zeroization as it's last powered action.

2.2.3 Neither snow nor rain nor heat…[1]

Orchestrated attacks to uncover iButton secrets by subjecting the iButton to extreme temperature or voltage conditions will generate a tamper response that results in zeroization. Deliberately exposure to temperatures outside the iButton's operational range of -20°C to +70°C (-4°F to +158°F) cause temperature monitors to trigger a cold-temp switch or high-temp effects that quickly zeroize to erase the contents of the

[1] "*Neither snow nor rain nor heat nor gloom of night stays these couriers from the swift completion of their appointed rounds.*" -- an inscription on the General Post Office, New York City. (see http://www.usps.gov/history/his8.htm)

memory. Voltages above or below maximum operating tolerances are clamped, and if excessive voltage is encountered, the I/O pin is designed to fuse and render the chip inoperable.

2.2.4 Fortresses large and microscopic…

In addition to these operation controls, the cryptographic chip is additionally protected. A substrate barricade is metallurgically- and glass epoxy-bonded to the active face of the chip. Attempts to remove the barrier to get to the chip cause a tamper response that results in zeroization. If a sophisticated attacker attempts to micro-probe the chip, they will encounter a shield of sub-micron pitch metal layers fabricated into a serpentine pattern directly on the chip. The chip will detect any break in this shield and immediately zeroize the chip.

2.3 DS1954B Firmware Capabilities

The Crypto iButton contains an 8051-compatible microcontroller, a tamper-evident real-time clock, a high-speed modular exponentiation accelerator for large integers up to 1024 bits in length, 32 Kbytes of ROM memory with preprogrammed firmware, 6 Kbytes of non-volatile RAM (NVRAM) for storage of critical data, input and output buffers with the standard iButton 1-Wire "front-end" for sending and receiving data, and control circuitry that enables the microcontroller to be powered up to interpret and act on the data placed in an input buffer, drawing its operating power from the 1-Wire line. The microcontroller, clock, memory, buffers, 1-Wire front-end, modular exponentiation accelerator, and control circuitry are integrated on a single silicon chip and packaged in a stainless steel case using packaging techniques which make it virtually impossible to probe the data in the NVRAM without destroying the data. Most of the NVRAM is available for use to support cryptographic applications such as those mentioned above.

The Crypto iButton firmware supports the Secure Hashing Algorithm (SHA-1) and conforms to Federal Information Processing Standard Publication (FIPS PUB) 180-1, *Secure Hash Standard (SHS)*.

2.4 Roles & Services

There are two separate roles in the operation of Crypto iButtons: Crypto Officer, and User. The Crypto iButton is intended to be activated by a service provider (Crypto Officer) who procures an iButton and co-issues the device to his own customers. These registered customers (Users) operate the devices as end users under a license agreement. The Crypto Officer loads the Crypto iButton with data to enable it to perform application specific functions. The User issues commands to the Crypto iButton to perform operations programmed by the Crypto Officer. For this reason the Crypto iButton offers functions to support the Crypto Officer in setting up the Crypto iButton for an intended

application, and it also offers functions to allow the authorized User to invoke the services offered by the Crypto Officer.

2.4.1 Transaction Groups, Objects and Scripts

A Crypto Officer can reserve a block of NVRAM memory to support services by creating a Transaction Group. A Transaction Group is simply a set of Objects that are defined by the Crypto Officer. These Objects include both data Objects (encryption keys, transaction counts, money amounts, date/time stamps, etc.) and Transaction Scripts which specify how to combine the data Objects in useful ways. The Crypto Officer creates a Transaction Group for a set of Users, which is independent of every other Transaction Group. Hence, a Crypto Officer could offer different sets of services in the same Crypto iButton using different Transaction Groups to support multiple functions for a single iButton. The number of independent Transaction Groups that can be supported depends on the number and size of the Objects defined in each Transaction Group. Examples of some of the Objects that can be defined within a Transaction Group are the following:

Modulus	Money	Register Input Data
Exponent	Clock Offset	Output Data
Transaction Script	Random Salt	Destructor
Transaction Counter	Configuration Data	

Within each Transaction Group, the Crypto iButton will initially accept certain commands that have an irreversible effect. Once any of these irreversible commands is executed in a Transaction Group, it remains in effect for the life of the Crypto iButton or until the Transaction Group to which it applies is erased from the Crypto iButton. In addition, there are certain commands that have an irreversible effect on the Crypto iButton as a whole, and remain in effect for the life of the Crypto iButton or until a master erase command is issued to erase the entire contents of the Crypto iButton. These commands are essential to give the Crypto Officer the necessary control over the operations that can be performed by the user. Examples of some of the irreversible changes made by a Crypto Officer are:

Privatize an Object	Lock an Object
Lock a Transaction Group	Lock the Crypto iButton

Since much of the Crypto iButton's utility centers on its ability to keep a secret, the Privatize command is the most important irreversible command. The fundamental concept implemented by the firmware is that the Crypto Officer can store Transaction Scripts in a Transaction Group to perform only those operations among Objects that he wishes the authorized User to be able to perform. The Crypto Officer can also store and privatize the private key or keys that allow the Crypto iButton to "sign" transactions on behalf of the Crypto Officer, thereby guaranteeing their authenticity. By privatizing and/or locking one or more Objects in the Transaction Group and locking the Transaction Group itself, the Crypto Officer maintains control over what the Crypto iButton is allowed to do on his behalf. After the group is locked the User cannot add new

Transaction Scripts to that Transaction Group and is therefore limited to the operations on Objects that can be performed with the Transaction Scripts programmed by the Crypto Officer.

2.4.2 Authentication

The Crypto iButton™ provides identification and authentication (I&A) functions for both role-based and identity-based I&A.

2.4.2.1 Identity-Based Authentication

The Crypto iButton™ can perform identity-based authentication using challenge-response protocols that use public key encryption or keyed message digest algorithms. Using these protocols, the iButton™ allows users to log in and log out, and provides automatic logout. When operated in the FIPS mode, the Crypto iButton™ employs identity-based authentication using challenge-response protocols with automatic logout. This advanced I&A is described in section 3.3, and may be used in conjunction with the role-based authentication described below.

2.4.2.2 Role-Based Authentication

The Crypto iButton™ may optionally use role-based authentication, in addition to or in lieu of identity-based I&A. There is a Crypto Officer personal identification number (PIN) for each iButton, which must be supplied with each and every service request reserved for the Crypto Officer. The Crypto Officer PIN (also called the iButton's common PIN) can be any value (numeric, alpha, or binary byte values), and is eight bytes in length. Similarly there is a User PIN for each Transaction Group, which must be supplied with every request for User services. There are also non-cryptographic services (related to iButton status) which are available to User and Crypto Officer without supplying an authenticating PIN.

2.4.3 Crypto Officer services

A Crypto Officer can exercise the following services with appropriate authentication:

SetCommonPIN	MasterErase
CreateTransactionGroup	LockCiB
DisableKeySetGeneration	

These allow the Crypto Officer to completely erase and zeroize an iButton, create new containers of User Objects and Scripts (Transaction Groups), prevent generation of additional keys, and lock the iButton to prevent additional groups from being added or changed. The Crypto Officer may also authenticate and assume a user role in order to set up information within a particular Transaction Group.

A complete description of each Crypto iButton command can be found in the Crypto iButton Firmware Reference Manual.

2.4.4 User services

A User can exercise the following services with appropriate authentication:

SetGroupPIN	CreateCiBObject
SetCiBObjectAttr	LockGroup
InvokeScript	ReadCiBObject
WriteCiBObject	DeleteGroup
ReadTrueTimeClock	ChangeGroupName
GenerateRSAKeySet	GenerateRSAModAndExp
GenerateRSAKeySetNP	GeneratePrime
GenerateRandomExponent	

These services allow a user to change PINs, create objects and scripts within a Transaction Group (subject to restrictions already set up by the Crypto Officer), create keys, read and write information from Objects, invoke Scripts, and modify Transaction Group characteristics. Changing a group name affects only name the label and not the Transaction Group ID; however, Locking a Transaction Group disables subsequent creation capabilities.

A complete description of each Crypto iButton command can be found in the Crypto iButton Firmware Reference Manual.

2.4.5 Status Functions

A number of status functions can be used to find the state of the iButton and various configuration information about the iButton. These status functions can be used by both User and Crypto officer without supplying any PIN:

ReadGroupName	GetGroupID
GetCiBConfiguration	ReadRealTimeClock
CheckGroupCRC	ReadRandomBytes
ReadFirmwareVersionID	ReadFreeRAM
GetCiBError	TestRNG

2.5 Key Management

The Crypto iButton has a unique internal 64-bit registration number which is not a key, is not private, and is also engraved on the outside of the module.

The iButton supports creation of RSA key pairs through the User functions GenerateRSAKeySet, GenerateRSAKeySetNP, GenerateRSAModAndExp and GenerateRandomExponent.

GenerateRSAKeySet generates a modulus, public exponent, and private exponent suitable for use as an RSA key pair. Modulus and Public exponent are locked (made read-only), and private exponent is privatized (only accessible by transaction scripts). GenerateRSAModAndExp performs the same function as GenerateRSAKeySet, but allows the public exponent to be chosen. GenerateRandomExponent generates a private exponent when modulus and public exponent are already defined

Hence, once keys are created the public keys cannot be modified, and only the scripts defined in the Transaction Group can use the private key. The private key can never be read. If the Transaction Group is locked, or the iButton is locked, the scripts cannot be changed.

GenerateRSAKeySetNP performs the same function as GenerateRSAKeySet but does not immediately privatize the private exponent. This would allow the newly generated private key to be read from the iButton until the private key Object is privatized. Although some applications may require this functionality, it is not recommended in the absence of strong procedural controls to protect the private key.

Keys are destroyed by deleting the Transaction Group that contains them (which destroys all objects within it, calling the master erase command (which destroys all Transaction Groups), or by triggering a tamper response.

3 Crypto iButton™ FIPS Mode

The Crypto iButton™ provides a rich set of cryptographic functionality in a physically secure package. This versatile module can be configured to function in a wide variety of applications such as an electronic change purse, biometric access token, or postage meter. The iButton™ can also be configured to operate in a FIPS 140-1 compliant mode. When configured and operated in this mode, the Crypto iButton™ provides a FIPS 140-1 level 3 compliant cryptographic module.

In all forms of operation, the Crypto iButton™ meets all level 3 and some level 4 FIPS 140-1 physical security requirements using sophisticated hardware security controls. Other areas of FIPS level 3 requirements can only be met if the Crypto iButton™ is configured for and operated in its FIPS mode.

3.1 FIPS Restrictions

FIPS 140-1 requires that RSA digital signatures follow the FIPS 186-1 standard. Currently, the Crypto iButton only support RSA digital signatures as specified in PKCS#1. Therefore, the Crypto iButton™ uses only SHA-1 for message digests in the FIPS mode.

The Crypto iButton™ provides several methods of identification and authentication to provide role-based or identity-based authentication. In order to meet level 3 FIPS 140-1 requirements the iButton™ must be operated using identity-based authentication. The module is operated as a single-user device with identity-based authentication of the user as described in section 2.4.2.1. The user immediately assumes both Crypto Officer and User roles. The authentication used in the FIPS mode incorporates a challenge-response protocol for login. The protocol ensures that no plaintext authentication data is transmitted over the 1-Wire™ interface. In addition, this authentication does not use RSA encryption, instead utilizing a keyed message digest algorithm incorporating SHA-1.

For operation in FIPS mode, no plaintext keys may be exported from the iButton™. Therefore, all keys are privatized and only accessed internally by scripts.

3.2 FIPS Configuration

To configure the iButton™ for operation in FIPS mode, the factory performs the following operations:

- Perform a Master Erase on the iButton™, removing all previous data.
- Initialize the common PIN to a random value known only to the factory. The common PIN is set with both the `PIN_TO_CREATE` and `PIN_TO_ERASE` options.
- Set a User login password.
- Add the User Transaction Group described in section 3.4.
- Lock the Transaction Group
- Create a final empty Transaction Group

- Lock the iButton to lock the final group, disable key generation, and prevent any future changes.
- Deliver the FIPS mode iButton™ and User login information.

3.3 Operation in FIPS mode

Operation of a Crypto iButton™ in FIPS mode is limited to a subset of the capabilities described in section 2.4. Because the factory initializes the button and does not release the Common PIN for the FIPS mode iButton™, the user cannot exercise any of the services listed in section 2.4.3.

In addition, the Transaction Groups and the iButton™ itself are locked, which prevents the User from exercising the following User services.

CreateCiBObject	SetCiBObjectAttr
LockGroup	DeleteGroup
GenerateRSAKeySet	GenerateRSAModAndExp
GenerateRSAKeySetNP	GeneratePrime
GenerateRandomExponent	

Thus, the User cannot create or delete Transaction Groups. Within existing Transaction Groups, the user cannot create new objects, cannot create new scripts, and cannot create new keys. The User is limited to exercising the status services and the following User services:

SetGroupPIN	ReadTrueTimeClock
ChangeGroupName	InvokeScript
ReadCiBObject	WriteCiBObject

The User can read the true time clock on the iButton, change the label attached to the User group, or change the Group PIN from its FIPS mode null default. In addition the user can use WriteCiBObject to write input data to a script (no other objects are writable), or ReadCiBObject to read the random login challenge, time for automatic logout, automatic logout interval, or output data from scripts.

The User can also invoke one of the four scripts: Login, Logout, EraseUser, and SHA1Digest. Three of these scripts are considered to be User role services (Login, Logout, and SHA1Digest) and one script is considered to be a Crypto Officer role service (EraseUser). The User must first login to the iButton™ using the Login script. To do this, the user reads the random challenge data. The random challenge is then "Xor"ed with the User login password and the result is hashed using SHA-1 to compute the challenge response. This response is provided to the Login script along with the number of seconds before automatic logout occurs.

Once a User has logged in, she may run the SHA1Digest script to hash data, run the Logout script, or await automatic logout. If the User runs the EraseUser script, or

provides ten successive invalid login responses, all User data will be erased from the iButton™ and the User deleted. Once a User has been erased, the User can no longer login and the FIPS mode iButton™ must be returned to the factory.

3.4 Factory Configuration Reference

All FIPS mode iButtons™ are delivered from the factory tested, operational, and configured. The Transaction Group described below containing the following scripts is loaded into the iButton™ customized with a User login password. The full factory configuration process is described in section 3.2.

3.4.1 FIPS Mode Transaction Group

The following transaction group is loaded into each FIPS mode iButton™. It includes four scripts described separately below, two writeable objects for input, and four readable objects for output. Only the Login script is available to the User before successful authentication. The Logout, EraseUser, and SHA1Digest scripts are destructible and are only available after login sets the destructor (LogoutTime) object.

```
{TransactionGroup for FIPS 140-1 Level 3 compliant identity based authentication}
TransactionGroup('FIPS Lev3 User1');

Begin
  Open:
    LoginInput:       InputData;     {A writable object for Login data input}
    SHAInput:         InputData;     {A writable object for SHA hashing input}

  Locked:
    RandomChallenge:  Configuration; {Login challenge string that changes every attempt}
    Timeout:          ClockOffset;   {How long to delay autologout}
    LogoutTime:       Destructor;    {Time when auto Logout occurs}
    Output:           OutputData;    {Readable output data}
    Login:            Script;
    Logout:           Script; Destructible;
    EraseUser:        Script; Destructible;
    SHA1Digest:       Script; Destructible;

  Private:
    Temp:             WorkingRegister; {Protected temporary calculation register}
    Response:         Money;           {Calculated response to login challenge}
    LoginPassword:    Configuration;   {User Login Password}
    FailedLoginCount: Counter;         {Unsuccessful login count}
    FailedLoginVal:   Money;           {Copy of FailedLoginCount for comparisons}
    ZeroVal:          Money;           {Literal constant}
    TenVal:           Money;           {Literal constant}
    ZeroOffset:       ClockOffset;     {Literal constant}
    NewChallenge:     Salt;            {New RandomChallenge value}
End
```

3.4.1.1 Login Script:

The login script accepts a response to the login from a User (operating in the User role) along with an automatic logout timeout interval. The response must be the SHA-1 hash of the challenge (from the RandomChallenge variable) "Xor"ed with the User login password. The automatic logout timeout interval defines how long the iButton™ remains logged in after the last successful Login or SHA1Digest script is executed.

The Login Script counts unsuccessful login attempts and will destroy the User data after 10 successive unsuccessful attempts. The random challenge is changed after every login attempt (whether or not it was successful).

```
Script Login;
{
LoginInput is a packet that contains two embedded objects.  The first
object is the response to the login challenge, and the second is the
number of seconds for the Crypto iButton to remain logged in (without
any script activity).

The Login script calculates the correct response to the challenge
using a keyed message digest and places it in the Response object.

If the input response matches the response generated by the script,
the seconds count passed in the packet is added to the current
value of the CiB's RTC (real time clock) and the result is placed
in the destructor object.

If ten login attempts fail, the user is prevented from logging in,
and all user information is destroyed.  The login challenge is changed
after every login attempt.
}

Begin
  {Make sure user has not been permanently erased }
  {When user is erased, FailedLoginVal has reached 10 }
  If FailedLoginVal = TenVal Then Begin
      Continue(EraseUser);  {Jump to erase user script}
      Exit(10);   {Return error code of 10 to indicate erased user}
  End

  {Compute correct response to challenge}
  Temp := LoginPassword Xor RandomChallenge;
  Response := SHA1(Temp);

  {Check computed response against input response}
  {The input response is the embedded Money object in LoginInput}
  If Response = LoginInput.Money[1] Then  {Login Succeeds}
    Begin
      FailedLoginCount := ZeroVal;  {Reset unsuccessful login count}
      FailedLoginVal   := ZeroVal;  {Reset unsuccessful login value}
      Timeout          := LoginInput.ClockOffset[1]; {Read logout delay from LoginInput}
      LogoutTime       := Timeout; {LogoutTime = Timeout + RTC} {Set logout delay}
      RandomChallenge  := NewChallenge;  {Generate new challenge}
      Exit(0);  {Successful login return code is zero}
    End

  {Login Fails}
  LogoutTime      := ZeroOffset; {disable login, record last attempt time}
  FailedLoginVal  := FailedLoginCount;  {increment failed login counter, store in value}
  RandomChallenge := NewChallenge;  {Generate new challenge}
  Exit(20); {Failed login return code is twenty}
End
```

3.4.1.2 Logout Script

The Logout Script immediately logs out a User without waiting for the automatic logout timer to take effect.

```
Script Logout;
{
Logout the User from the transaction group before the timeout logs you out automatically.
}

Begin
  LogoutTime := ZeroOffset;  {logout by setting destructor to current time}
End
```

3.4.1.3 SHA-1 Digest Script

The SHA-1 Digest Script allows a User (operating in the User role) to hash input data and receive the message digest in response. Performing a hash also resets the automatic logout timer to the timeout value specified by the user at login.

```
Script SHA1Digest;
{
Perform a SHA1 message digest on an input value.
The output is returned in the Output variable.
}

Begin
  {SHA-1 hash the input and return it in the Output}
  Output := SHA1(SHAInput);

  {reset logout timer because of this activity}
  LogoutTime := Timeout; {LogoutTime = Timeout + RTC}
End
```

3.4.1.4 Erase User Script

The Erase User Script allows a User (operating in the Crypto Officer role) to actively erase his User information. This effectively destroys all critical security parameters stored in a FIPS mode iButton. This script is also effectively run if there are ten successive unsuccessful login attempts.

```
Script EraseUser;
{
Erase a user's authentication data and prevent that user from
logging in ever again.  This script is to provide a user a
local zeroization function within a single Transaction Group.
}
Begin
  LoginPassword := ZeroVal;   {Destroy user secret value}
  RandomChallenge := ZeroVal; {Destroy login challenge value}
  TimeOut := ZeroVal;         {Reset inactivity time value}
  LogoutTime := ZeroOffset;   {Log user out for last time}
  FailedLoginCount := TenVal; {Never allow user to login again}
  FailedLoginVal := TenVal;   {Never allow user to login again}
  Exit(10);   {Return error code of 10 to indicate erased user}
End
```

3.4.2 FIPS Mode Symbol File

Every iButton™ Transaction Group requires a symbol file, which assigns internal identifiers and initial values to all objects. The FIPS mode symbol file sets an initial random value for the RandomChallenge and sets the User login password.

```
    LoginInput          =$01  {+ S$1C -}
    RandomChallenge     =$02  {+ S$80 I(R$80) -}
    Timeout             =$03  {+ S$04 -}
    LogoutTime          =$04  {+ S$04 I($00) -}
    Login               =$05
    Logout              =$06
    EraseUser           =$07
    SHA1Digest          =$08
    LoginPassword       =$09  {+ S128 I'Any password can be set here' -}
    FailedLoginCount    =$0A  {+ S1 I($00)-}
    FailedLoginVal      =$0B  {+ S1 I($00)-}
    ZeroVal             =$0C  {+ S1 I($00)-}
    TenVal              =$0D  {+ S1 I($0A)-}
    NewChallenge        =$0E  {+ S$80 I(R$80) -}
    Response            =$0F  {+ S$14 -}
    ZeroOffset          =$10  {+ S$04 I($00) -}
```

```
    SHAInput            =$11  {+ S$FF -}

{Auto Objects}
    Output              =$A0
    Temp                =$A2
Functions:
    SHA1                =$01
```

End of document.