A Practical Introduction to the Dallas Semiconductor iButton™

Kingpin

@Stake, Inc.
One Kendall Square, Building 200, 2nd Floor, Cambridge, MA 02139, USA.
http://www.atstake.com
E-mail: kingpin@atstake.com

Abstract. The Dallas Semiconductor iButton™ is one of a new breed of devices that can be used as hardware tokens in the PKI arena. This paper introduces and gives an overview to the iButton™ family, focusing on the hardware component, and examines potential areas that may be susceptible to attack. These areas may yield vulnerabilites or problems related to the secureness of the device. Detail is placed on the Java™-powered cryptographic iButton™, which plays a major role in the implementation of PKI or other solutions where security and/or encryption is needed. Simple software routines to communicate with the iButton™ are also presented.

Keywords: iButton, hardware token, PKI, Java, attack methods

1 Introduction to the iButton™

The Dallas Semiconductor iButton™ is a unique device meant to replace smartcards, magnetic stripe cards, barcodes, and radio-frequency proximity cards (RFID) for use in access control, cashless transactions, PKI, authentication, identification, Internet commerce and many other solutions that require portability and security [5].

The physical device is a 16mm dime-sized metal can (Figure 1). There are a number of advantages to the iButton™ compared to previously used hardware token devices:

- **Rugged.** The iButton™ is housed in a water-proof, stainless steel metal housing. Due to the housing armor, the iButton™ can withstand extreme environmental conditions and handling with no loss of data or performance. The device has been wear-tested for 10-year durability. The ruggedness of the device is an extreme advantage over smartcards and other devices that contain only minimal circuitry protection. For example, smartcard contacts are inherently fragile and any mishandling often leads to wirebond breakage and irrepairable damage.

- **Wearable.** Unlike credit card-sized smartcards and large keychain-sized USB keys and other hardware tokens, the iButton™ is small enough to mount onto wearable accessories.

- **Tamper Responsive.** The iButton™ has been touted as having a number of tamperproofing features which prevent the device from being physically attacked with invasive methods. A later section describes these specific features.

Figure 1: Typical appearance of an iButton™. [3]

The family of iButtons™ is split into two sections:

- **Touch Memory iButtons™** make up a large portion of the family. These devices can serve as replacements for stored-value or debit card applications, the majority of which are currently handled by smartcards.

- **Java™-powered Cryptographic iButtons™** consist of a microprocessor and high-speed math accelerator, the foundations for a complete cryptographic engine. The device also runs a Java Card 2.0-compliant Java Virtual Machine.

2 Touch Memory iButtons™

The first generation of the iButton™ family consist of the Touch Memory devices. These devices are often used for very specific implementations. Depending on the button type, a variety of application-specific features are provided [6]:

- Up to 64Kbit of One-Time Programmable Read-Only Memory
- Up to 64Kbit of Non-Volatile RAM
- Temperature Sensor
- Real-Time Clock

Any type of digital data can be stored in these buttons. Unfortunately, none of the Touch Memory iButtons™, with the exception of the Monetary, Crypto, and Multikey iButtons™, employ any type of security protection to limit access to the data stored on the button.

3 Java™-Powered Cryptographic iButton™

Often considered the second generation of the iButton™ family is the Java™-Powered Cryptographic iButton™, also referred to as the DS1954. The device is a feature-rich iButton™ with a focus on e-commerce and secure transactions. Although the iButtons™ can be mounted in a variety of fashions, the Java™-Powered Cryptographic iButton™ is most often crafted into a ring (Figure 2).

Figure 2: Typical appearance of the Java™-powered cryptographic ring. [3]

The Java™-Powered Cryptographic iButton™ has a feature set that mimics, and often betters, that which is supported by smartcards and other hardware token devices [14, 15]:

- **8051-compatible microcontroller.** The Dallas Semiconductor DS83C950 security processor was specifically designed for security-concious applications.

- **High-speed math accelerator.** Designed for 1024-bit public key cryptography.

- **Large ROM/RAM configuration.** 6kB of Non-Volatile RAM and up to 64kB ROM allow sufficient memory space for the pre-loaded Java™ Virtual Machine (JVM) and execution of multiple Java™ applets.

- **Java Card™ 2.0-compliant.**

- **Pending FIPS 140-1 security certification.** Designed for compliance to the U.S. government standards for cryptographic devices.

By designing the iButton™ with hardware cryptographic support and by allowing the device to execute Java™ applications, a number of security solutions can be implemented. The tamperproofing features of the iButton™ are designed to prevent intruders from accessing critical data and obtaining private keys from the device.

The physical design of the device is aimed to be responsive to tampering. By viewing the internal detail of the Cryptographic iButton™ (Figures 3, 4), some of the attempts at tamper detection are evident. The two microswitches could potentially be bypassed or replaced as to avoid tamper detection. The 32.768kHz quartz crystal is used to control the real-time clock, not the whole system. However, manupulating this signal might have some attack benefits. Attack mechanisms are discussed in a later section of the article.

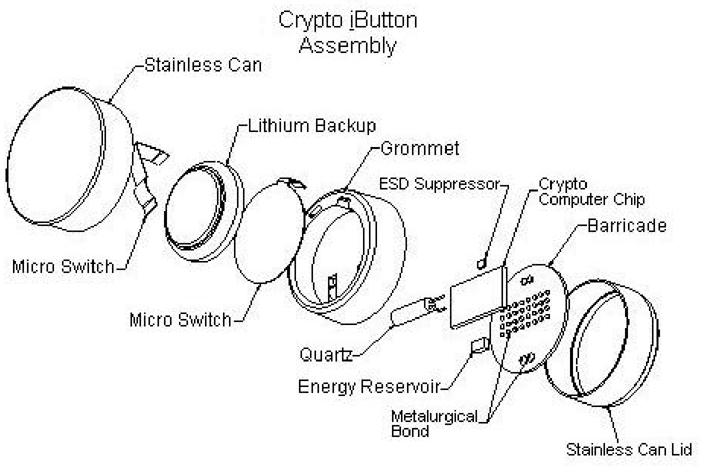

Figure 3: Assembly detail of the cryptographic ring. [15]

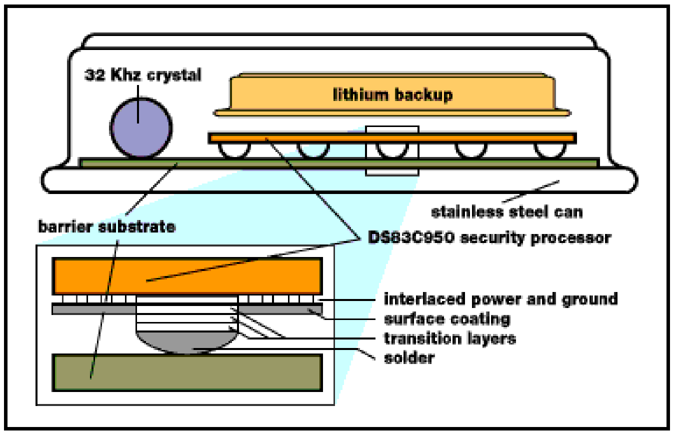

Figure 4: Internal construction detail of the cryptographic ring. [13]

4 Low-Level Functionality

4.1 Identification

All the iButtons™ have the same underlying technology for communication using the novel Dallas 1-Wire™ interface. The data is both read and written with a single I/O pin plus signal ground. By toggling the direction of a port pin (input or output) on a microprocessor, one can transmit commands, serially, bit by bit, to the iButton™ and read its responses.

By using minimal electronic circuitry, often just a pull-up resistor and a Zener or Transient Voltage Suppressing diode for port pin protection from static discharge, one can easily interface the iButton™ with a microprocessor or host PC. The internal circuitry of the iButton™ lends itself to easy, albeit timing-sensitive, communications.

Each iButton™, no matter what type, is assigned a 64-bit unique ID etched into the silicon and engraved on the stainless steel housing (Figure 5). This number can be the credential for a basic security scheme and act as a low-cost electronic key.

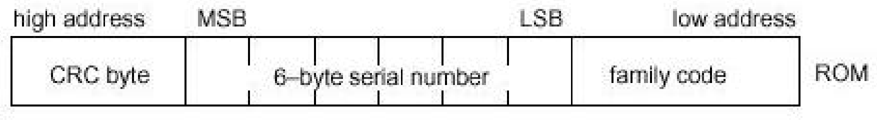

Figure 5: 64-bit ID unique to each iButton™. [9]

- **Family Code.** The 1-byte family code identifies the specific type of iButton™.

- **Serial Number.** The 6-byte serial number is said by Dallas Semiconductor to be unique and un-alterable. Various potential attacks could be mounted using the "uniqueness" of a device, which are discussed in a later section.

- **CRC.** The 1-byte Cyclic Redundancy Check (CRC) is derived from the value of the first 56 bits [10]. This can and should be used by the host system to verify proper data transfer.

Currently, the 64-bit unique ID is not secret and is printed directly onto the stainless steel housing of the iButton™. Other information is also displayed, including date stamp, package revision, and package type (Figure 6). If the security implementation is based solely on the ID itself, displaying the unique ID in plain view is not advisable and may lead to cloning or spoofing attacks. In addition, past experience has shown that people will use their readily accessible value as data input to the cryptographic keying process. For example, Dallas Semiconductor recommends "using the never duplicated number to seed an encryption algorithm" [11]. Lax implementations of encryption algorithms lead to extremely weak security systems.

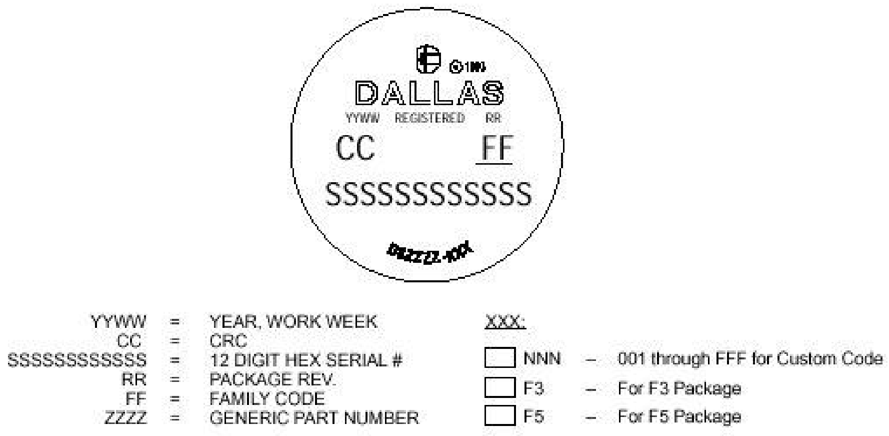

Figure 6: iButton™ engraving on stainless steel housing. [4]

4.2 Communication

The 1-Wire™ interface used to communicate with the iButton™ is a strictly defined serial protocol designed by Dallas Semiconductors. Communications with the iButton™ are extremely timing sensitive and half-duplex, meaning only one device may transmit or receive at any given moment, but not both. The 1-Wire™ protocol is based on a master/slave configuration, in which the host PC or microprocessor is the master and the iButton™ device is the slave. Multiple iButton™ devices are capable of being on the 1-Wire™ bus at any given time.

The transfer of information between the host and the iButton™ is based on a time slot, t, with a period between 60uS and 120uS. Within this time slot, a "zero" is defined as a low pulse for the whole time, and a "one" is defined as a low pulse for up to 15uS, then remaining high for the remainder of the time slot [12].

Two software routines are used for low-level communication between the host and the iButton™. Source code is supplied for Motorola 68000-series processors.

- **TouchReset** will check for the presense of an iButton™ and send a required Reset pulse to prepare it for data communication. This routine will also check for a short circuit of the 1-Wire™ interface.

- **TouchByte** is the core data transfer routine. Specific iButton™ commands are sent using this routine and the response from the iButton™ device is returned.

4.2.1 TouchReset

This procedure transmits the Reset signal, a 480uS low pulse, to the iButton™ device and watches for a returned Presense Pulse (Figure 7). When the iButton™ is inserted into its socket, power is supplied by the 1-Wire™ interface. A Presence Pulse is initially generated by the iButton™ to signify to the host that it has been connected and is now ready for use. When the processor detects the initial Presense Pulse from the iButton™, it responds with the TouchReset sequence, which acts as an acknowledgement and verification that the iButton™ is ready for data transfer.

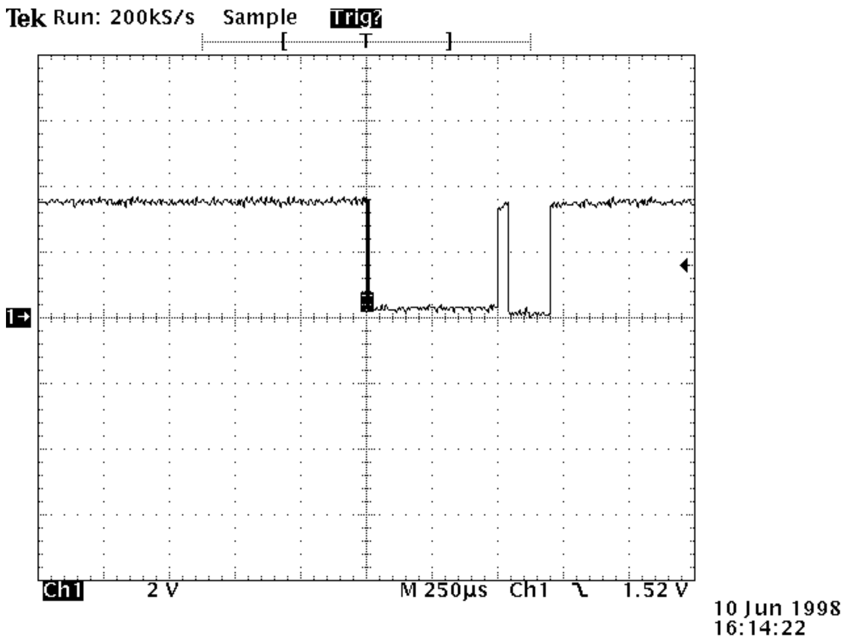

Figure 7: iButton™ Reset Pulse and returned Presence Pulse.

```
int TouchReset(void)
{
  int i=0;

  // Reset pulse = 480uS (7958 cycles)

  *PK_DIR |= PK4_OUTPUT;    // Pull pin low (16 cycles)

  asm(
  "    MOVE.L #304,D0        ; 12 cycles",
  "    DBEQ D0,*             ; 10 cycles if taken, 14 if not"
  );

  *PK_DIR &= PK4_INPUT;     // Pull pin high (18 cycles)

  asm(
  "    MOVE.L _PK_DATA,A0",
  "    MOVE.L #1393,D0       ; Set 3360uS timer (40 cycles/loop)",
  "WAITLOW:  BTST #3,(A0)   ; Check pin - 10",
  "    BNE    WH             ; Exit loop if line is high (Z=0)-12",
  "    TST    D0             ; Set condition codes for D0 - 8",
  "    DBEQ   D0,WAITLOW     ; Wait to see if line will go high-10",
  "    MOVE.B #1,{i}         ; Set i=1 if short circuit",
  "    BRA    SHORT          ; If line is still low, must be short
                             ; circuit",
  "WH: MOVE.L #153,D0        ; Set 480uS timer for presence detect
                             ; (52 cycles/loop)",
  "HL: BTST   #3,(A0)        ; Check pin - 10",
  "    SEQ    D1             ; Put $FF in D1 if low pulse (Z=1) is
                             ; detected - 12",
  "    OR.B   D1,{i}         ; If a low pulse was found, set {i} –
                             ; 12",
  "    TST    D0             ; Set condition codes for D0 - 8",
  "    DBEQ   D0,HL          ; Wait for entire 480uS Master RX slot
                             ; - 10",
  "SHORT:                    ; End"
  );

  if (i == 255) // $FF
  {
      // Presence Pulse detected
  }
  else if (i == 1)
  {
      // Short circuit of 1-Wire interface
  }
  else // i == 0
  {
      // No Presence Pulse detected
  }
}
```

4.2.2 TouchByte

This procedure is the core data transfer routine which transmits and receives iButton™ commands and responses. The procedure sends a one byte long command to the iButton™ device, bit by bit, and subsequently receives a one byte long response from the iButton™. Sending and receiving specific commands using this routine will allow complete control of the Touch Memory iButtons™.

TouchByte consists of a loop that transfers a single bit of information at a time between the host and the iButton™. A single I/O port pin, consistant with the 1-Wire™ interface protocol, is used to both send and receive the data. Setting the port pin as either an input or output will determine the logic levels of the 1-Wire™ interface and affect how the data is interpreted. The state of the port pin is varied many times during a data transfer.

```
unsigned char TouchByte(unsigned char outch)
{
   int i,k;
   unsigned char inch = 0, databit; // Input byte received from button
   // BITLOOP:
   // Tlow0 = 71uS - 75uS
   // Tlow1 = 11uS

   for (k = 0; k < 8; ++k) // Setup to transmit/receive 8 bits
   {
     databit = (outch >> k) & 0x1; // Shift outch to correspond to
                                   // correct bit

     *PK_DIR |= PK4_OUTPUT;  // Pull pin low (16 cycles)

     // Make sure Touch Memory sees a low for at least 1uS (16
     // cycles) but not more than 15uS (248 cycles)
     asm(
     "  MOVE.L #2,D0  ; 12 cycles",
     "  DBEQ   D0,*   ; 10 cycles if taken, 14 if not"
     );

     // If databit = 1, set DIR to INPUT for remaining time in slot
     // (60uS min.) = 0, keep DIR as OUTPUT for remaining time in slot
     // (Tlow=70uS)
     if (databit == 1)
     {
        *PK_DIR &= PK4_INPUT; // 18 cycles

        // Delay 5uS (83 cycles) to give the data returned from Touch
        // Memory time to settle before we read it.
        asm(
        "      MOVE.L #3,D0   ; 12 cycles",
        "      DBEQ   D0,*    ; 10 cycles if taken, 14 if not",
        "       MOVE.L _PK_DATA,A0          ",
        "       BTST   #3,(A0)       ; Sample data line",
        "       SNE    {i}           ; i = $FF if received bit = 1"
        );

        // Get bit from touch memory
        inch = inch | ((i & 0x1) << k); // Shift in bits (LSB first)
      }

      // Wait out remining time slot (60uS min. = 995 cycles)
      asm(
      " MOVE.L #37,D0         ; 12 cycles",
      " DBEQ D0,*             ; 10 cycles if taken, 14 if not"
      );

      *PK_DIR &= PK4_INPUT;  // Pull pin high (18 cycles)
  }

  return (inch);
}
```

The standard iButton™ command set allows a decent range of functionality and allows low-level interaction with the iButton™. The commands are divided into two groups. A subset of each group is listed below [1]:

- **ROM Commands** make use of the 64-bit unique ID for purposes of device identification and broadcasting.

  READ ROM. To identify a 1-Wire™ compliant device. iButton™ will respond with its 64-bit unique ID.

  SKIP ROM. To broadcast data to all devices connected to the bus.

  MATCH ROM. To send data to a specific device on the bus.

  SEARCH ROM. To identify all 1-Wire™ compliant devices on the bus. Each iButton™ will respond with its 64-bit unique ID.

- **Advanced Commands** are specific commands related to a particular iButton™ device function set, making use of on-button memory, real-time clock, or temperature sensor modules.

  READ MEMORY. To read one or more consecutive bytes of information from the iButton™ given a valid starting address.

  READ/WRITE SUBKEY. To read/write one or more consecutive bytes of information from/to a password-protected page.

READ/WRITE SCRATCHPAD. To read/write one or more consecutive bytes of information from/to the scratchpad (temporary storage) area.

COPY SCRATCHPAD. Move the data within the entire scratchpad area into a password-protected page. Automatically erases scratchpad area when transaction is complete.

The TMEX (Touch Memory EXecutive™) API by Dallas Semiconductor provides a layer of abstraction between a user's application code and low-level routines, which may be useful depending on the customer's requirements. In addition, when writing Java™ applets for the Java™-Powered Cryptographic iButton™, much of the low-level functionality is performed by the Java™ Virtual Machine, allowing the user to focus on application-level code.

5 Potential Areas of Attack

Many of the attacks possible to exercise on the iButton™ are common to other embedded system attacks. Kömmerling/Kuhn have defined four high-level attack categories which were used in smartcard analysis and could be duplicated with the iButton™ [2]:

- **Microprobing** consists of invasive techniques used to access the device internals directly.
- **Software attacks** rely on the normal communications interface of the device and, by having the device execute custom software, exploit security vulnerabilities.
- **Eavesdropping** techniques monitor the external connections to the device and any stray EMI/RF noise radiated from the device during normal operation.
- **Fault generation** techniques use abnormal environmental conditions to intentionally cause failures in the device. By doing so, the device may function outside of its intended feature set.

Differential Power Analysis cannot readily be used, unless invasive methods are successful in accessing the iButton™ internal battery power supply rail. However, it might be useful to analyze the current consumption and characteristics of the 1-Wire™ interface.

5.1 Invasive Attacks

Invasive attacks require access to the physical device and are often destructive. Depending on the complexity of the device, special laboratory techniques and chemicals might be used. An example would be the depackaging of smartcards [2], in which hot fuming nitric acid is used to gain access to the die.

Many devices contain tamperproofing mechanisms against invasive attacks, which will render the device inoperable if tampering is detected. In order for invasive attacks to be successful, access to the devices internals must be completed without detection.

5.1.1 Uniqueness

The 64-bit identification, comprising the family code, serial number, and CRC is guaranteed by Dallas Semiconductor to be unique. Systems designed solely around uniqueness often have problems. Two examples are the cellular phone system and ethernet MAC addresses.

The cellular phone system is designed around the premise of a unique electronic serial number (ESN) and mobile identification number (MIN) pair for each user [7]. If an ESN/MIN pair is cloned and entered into another phone, security is nil and the telephone system will not be able to detect the difference between the actual legitimate user and the cloned phone.

Ethernet MAC addresses are trivial to clone in both hardware and software, allowing one to bypass copyright protection and launch denial-of-service and race condition attacks.

It might be possible to clone the iButton™ by modifying the unique ID by invasive or non-invasive techniques. If the 48-bit unique serial number is changed, the CRC must also be changed, since the value will now be incorrect. The 64-bit unique ID is marked onto the iButton™ in two different locations:

- **Stainless Steel Housing.** The proposed methods of attack in this paper focus on the electrical properties and system functionality of the device, so the etchings on the physical iButton™ housing need not be considered.

- **Internal ROM.** The ID is created by selectively removing 3-micron polysilicon links, each which define a "zero" or "one". The links are sealed with a protective layer of glass, so that any tampering attempts would be evident [11]. It may be possible to vary the ID by opening closed links or reforming open links. If the device still functions and the particular attack is successful, having physical evidence of tamperproofing, such as the broken glass layer, is a moot point.

5.1.2 NVRAM Access

A common attack, fitting into the Microprobing group, is to gain physical access to the pin connections or wire bonds of the internal memory of a device and dump all memory locations. This is done in hopes of obtaining critical data, encryption keys or other information that is considered to be secure.

The Java™-Powered Cryptographic iButton™ includes a tamper-detection feature that will immediately erase all internal NVRAM when the physical perimeter of the device is compromised.

5.1.3 Processor Clock Skewing

Gaining external control of the clock signal of the system can be a valuable tool.

- **Increasing clock speed** will allow the attacker to view more iterations of calculations or look for repeated sequences. This attack is useful when analyzing time-based hardware token devices.

- **Decreasing clock speed** will allow the attacker to slow down or single-step through operations and analyze the data using external measurement tools.

- **Halting system execution** at a known point will allow the attacker to repeatedly examine a particular operating condition more closely. This is done by applying the same number of clock cycles each time after reset.

- **Inducing calculation errors** is often possible by inducing abnormal clock signals into the system. Calculation or execution errors may lead to conditions where critical information is leaked or incorrectly handled.

The processor internal to the Java™-Powered Cryptographic iButton™ is driven by an unstabilized oscillator ranging from 10 to 20MHz [13]. By having the iButton™ vary its clock frequency, it makes clock skewing attacks more difficult.

5.2 Non-Invasive and Software Attacks

With non-invasive attacks, the physical device is not harmed or tampered with. Non-invasive and software attacks often, but not always, make use of the normal operating conditions of the device. Once the attack is designed and successful, the results are extremely reproducable from one device to another.

Many of the software attacks on the Java™-Powered Cryptographic iButton™ are launched through the 1-Wire™ interface, due to the fact that it is the only communications path. A Java™ program would be loaded into the device using the normal communications means, and executed normally as a Java™ applet. The possibility of software attacks are great and the area should be investigated thoroughly. There are specific Java™ related problems that could be used to help launch an attack, including byte code verification, problems with code signing, and resource starvation. Any Java™ applet loaded onto the Java™-Powered Cryptographic iButton™ can be a target for attack.

5.2.1 Spoofing

In the same vein of the invasive uniqueness attacks, a non-invasive method of spoofing of the 1-Wire™ communication signal is entirely feasible. By monitoring the communication stream of the Touch Memory or Java™-Powered Cryptographic iButton™ during normal usage, secret key or other critical information may unknowingly be transmitted.

Imitating an iButton™ with electronic circuitry to clone a device would also be possible. Full device "emulation" would be easier accomplished with the low-featured Touch Memory iButtons™. Depending on the system implmentation, it would be trivial to imitate an iButton™ and transmit its cloned 64-bit unique ID to the host PC in hopes of the host transmitting and revealing critical data.

5.2.2 Misuse of Memory Management

If hardware or software memory management is unused, misused, or misconfigured, it may be possible to access protected memory and scratchpad areas that are considered critical. Due to programming errors of particular Java™ applications running on the iButton™, critical data or secret keys might not be cleared from memory, thus giving the attacker information that may be used in future attacks. The attacking software application would attempt to dump all memory areas and access any memory-mapped peripherals. The Kerberos 4 exploit takes advantage of such a situation, in which the username and Kerboros realm of the previously logged in user was recovered due to improper clearing of critical memory [8].

6 Conclusions

Choosing the iButton™ over other hardware token devices for system implementation is dependent on the customer's specific needs. The iButton™ is definetely the most robust of the existing hardware token devices. The tamper detection and security features of both the Touch Memory and Java™-Powered Cryptographic iButton™ make them useful for implementations where physical attack may be commonplace.

The Java™-Powered Cryptographic iButton™ is an attractive solution for companies that would like to execute custom Java™ applications on-the-fly or on-button for security purposes. Although the iButton™ appears to be sufficiently secure for many applications, care must still be taken to assure that the implementation surrounding the device is also secure.

The fact that the iButton™ has not been widely deployed in the United States might make clients, who are used to smartcards, feel uncomfortable. The iButton™ is used extensively in other coutries, and supporting data of the success of the product can be found on the Dallas Semiconductor web page [3].

Further research will be done to attack the assumed security of the Java™-Powered Cryptographic iButton™, which is the most directed for use in Internet commerce applications.

7 Additional Reading

1. R. Anderson and M. Kuhn, "Tamper Resistance – a Cautionary Note," *The Second USENIX Workshop on Electronic Commerce Proceedings*, November 1996, http://www.cl.cam.ac.uk/ftp/users/rja14/tamper.ps.gz.
2. R. Anderson and M. Kuhn, "Low Cost Attacks on Tamper Resistant Devices," *Security Protocol Workshop*, April 1997, http://www.cl.cam.ac.uk/ftp/users/rja14/tamper2.ps.gz.
3. P. Kocher, J. Jaffe, and B. Jun, "Differential Power Analysis", http://www.cryptography.com/dpa/index.html.
4. R. Di Giorgio, "iButtons: The first ready-to-buy 2.0 Java Card API devices", *JavaWorld*, April 1998, http://www.javaworld.com/javaworld/jw-04-1998/jw-04-ibuttons_p.html.
5. C. McManis, "My ENIGMAtic Java Ring", *JavaWorld*, August 1998, http://www.javaworld.com/javaworld/jw-08-1998/jw-08-indepth_p.html.
6. Z. Chen, "Understanding Java Card 2.0", *JavaWorld*, March 1998, http://www.javaworld.com/javaworld/jw-03-1998/jw-03-javadev.html.
7. Sun Microsystems Java Card™ Technology Web Page, http://java.sun.com/products/javacard/.
8. Java-Powered Cryptographic iButton™: Software Development Tools and Documentation Web Page, http://www.ibutton.com/jibkit/index.html.

References

1. Dallas Semiconductor, "Book of DS19xx iButton Standards", 1995, pp. 45-46., http://www.ibutton.com/ibuttons/standard.pdf.
2. O. Kömmerling and M. Kuhn, "Design Principles for Tamper-Resistant Smartcard Processors," *USENIX Workshop on Smartcard Technology Proceedings*, May 1999.
3. Dallas Semiconductor iButton™ Web Page, http://www.ibutton.com.
4. Dallas Semiconductor, "Book of DS19xx iButton Standards", 1995, pg. 26., http://www.ibutton.com/ibuttons/standard.pdf.
5. Dallas Semiconductor iButton™ Applications Web Page, http://www.ibutton.com/applications/index.html.
6. Dallas Semiconductor Memory iButtons™ Web Page, http://www.ibutton.com/ibuttons/memory.html.
7. B. Oblivion, "Cellular Telephony: Part 1", http://www3.L0pht.com/~oblivion/blkcrwl/cell/texts/cell-rdt.txt.
8. Mudge, "Kerberos 4 Advisory", *L0pht Heavy Industries Security Advisory*, November 1996, http://www.L0pht.com/advisories/krb_adv.html.
9. Dallas Semiconductor, "Book of DS19xx iButton Standards", 1995, pg. 14., http://www.ibutton.com/ibuttons/standard.pdf.
10. Dallas Semiconductor, "Book of DS19xx iButton Standards", 1995, pg. 126., http://www.ibutton.com/ibuttons/standard.pdf.
11. Dallas Semiconductor, "Automatic Identification Data Book", 1995, pg. xxx.
12. Dallas Semiconductor, "Reading and Writing iButtons™ via Serial Interfaces", *Application Note #74*, pp. 2-6, http://www.dalsemi.com/DocControl/PDFs/app74.pdf.
13. S. Curry, "An introduction to the Java Ring", *JavaWorld*, April 1998, http://www.javaworld.com/javaworld/jw-04-1998/jw-04-javadev_p.html.
14. Dallas Semiconductor Java-Powered™ Ring Fact Sheet Web Page, http://www.ibutton.com/store/jringfacts.html.
15. DS1954 Cryptographic iButton™ FIPS 140-1 Non-Proprietary Cryptographic Module Security Policy Web Page, http://www.ibutton.com/software/crypto/fips140-112/.