Practical Secure Hardware Design for Embedded Systems

Joe Grand*
Grand Idea Studio, Inc.

ABSTRACT

The design of secure hardware is often overlooked in the product development lifecycle, leaving many devices vulnerable to hacker attacks resulting in theft of service, loss of revenue, or a damaged reputation. Many times, products must be redesigned after a harmful incident, which raises overall development costs and increases time-to-market. This paper focuses on general concepts for secure hardware design coupled with practical examples. Topics in this paper include recommendations on incorporating security into the product development cycle, attack and threat models, and design solutions for enclosure, circuit board, and firmware layers.

Keywords: secure hardware, embedded systems, computer security, hacker attacks

1. INTRODUCTION

The primary goal of this paper is to introduce the reader to the concepts of designing secure hardware in embedded systems. Understanding the major classes of attack and the mindset of potential attackers will go a long way in helping one to decide on the best and proper secure hardware design methods for a particular application. Examples of previous hardware attacks are discussed throughout the paper. By learning from prior attacks, we can understand the weaknesses and mistakes of such designs and improve upon them in our own products. We also provide numerous references and resources for the reader to explore in more detail.

Section 2 explores the typical product development cycle and recommends ways to incorporate security, risk assessment, and policies into the process. Section 3 defines a baseline classification of attackers, attack types, and threat vectors. Section 4 explores the practical design solutions for the enclosure, circuit board, and firmware layers of a product.

2. SECURITY IN THE PRODUCT DEVELOPMENT CYCLE

As designers, the best we can do is understand the potential attacks against our system, design methods to prevent such attacks, with the understanding that nothing is ever 100% secure. "Secure" can simply be defined as when the time and money required to break the product is greater than the benefits to be derived from the effort. Given enough determination, time, and resources, an attacker can break any system. Security is a process, not a product. Security must be designed into the product during the conceptual design phase and must be considered for every portion of the design. It must be continually monitored and updated in order to have the maximum effect against attacks. Security cannot be simply added to a product and forgotten about, assuming that the product will forever remain secure.

While many design methodologies exist, the primary concern is to incorporate risk analysis and security considerations into each step of the cycle. Having high-level processes in place will help to ensure that the low-level design details are properly implemented. In NIST's *Engineering Principles for Information Technology Security (A Baseline for Achieving Security)* [33], a number of security principles are provided that can be applied to any design process:

* joe@grandideastudio.com, http://www.grandideastudio.com

† This paper originally published by CMP Media in the *Proceedings of the 2004 Embedded Systems Conference*, San Francisco, California, March 29-April 1, 2004.
† Last updated on June 23, 2004.

1. **Establish a sound security policy as the "foundation" for design.** The security policy identifies security goals the product should support (see Section 2.1 for details). The goals guide the procedures, standards, and controls of the development cycle. It may be necessary to modify or adjust security goals due to other operational requirements.

2. **Treat security as an integral part of the overall system design.** Security must be considered during product design. It is very difficult to implement security measures properly and successfully after a system has been developed.

3. **Reduce risk to an acceptable level.** Risk is defined as the combination of the probability that a particular threat source will exploit a vulnerability and the resulting impact should this occur. Elimination of all risk is not cost-effective and likely not possible. A cost-benefit analysis should be conducted for each proposed secure hardware mechanism.

4. **Implement layered security (Ensure no single point of failure).** Security designs should consider a layered approach of multiple security mechanisms to protect against a specific threat or to reduce a vulnerability.

5. **Strive for simplicity.** The more complex the mechanism, the more likely it may possess exploitable flaws. Simple mechanisms tend to have fewer exploitable flaws and require less maintenance.

6. **Minimize the system elements to be trusted.** Security measures include people, operations, and technology. Where technology is used, hardware, firmware, and software should be designed so that a minimum number of elements need to be trusted in order to maintain protection. In Kocher's *Hacking Cryptosystems* presentation [24], it is recommended to "put all your eggs in one basket" by isolating all critical content into one secure area instead of having multiple secure areas throughout the design. This way, you can focus on properly securing and testing a single critical area of the product instead of many disparate areas.

7. **Do not implement unnecessary security mechanisms.** Every security mechanism should support one or more defined goals. Extra measures should not be implemented if they do not support a goal, as they could add unneeded complexity to the system and are potential sources of additional vulnerabilities. All likely classes of attack should be protected against (see Section 3 for details).

Many times, an engineering change will be made to the product circuitry or firmware without re-evaluating the effect such a change may have on system security. Without a process in place to analyze changes throughout the design cycle, security that was properly implemented at the beginning of the design may become irrelevant by the time the product goes into production. Requiring trusted third-party design reviews of the product during the prototype and pre-production phases allow a "fresh set of eyes" to examine the product for any critical design flaws that are non-obvious or have been simply overlooked by the product designers.

2.1. Risk Assessment and Management

Choosing what type of security methods to use to protect your embedded intellectual property is no different than choosing what type of safe to use to protect your valuables. A priceless family heirloom may likely be stored in a torch-, explosive-, and tool-resistant safe. However, it would not make much financial sense to purchase such a safe to store an easily replaceable piece of jewelry. By the same token, it would not be feasible to implement an extremely secure, multi-layered hardware solution just to protect a single password that is used to access undocumented games in a mobile phone, although it would be in order to protect cryptographic keys used for encrypted communications with a large financial institution (whose theft could result in millions of dollars of loss). Before being able to make an educated, informed decision, one needs to understand the threat, the value of the contents being protected, and the reason for protecting such contents. Essentially, weaker, more vulnerable devices should contain less valuable secrets.

Before deciding on acceptable secure hardware methods to design into your product, risk assessment of three key areas must take place:

- **What needs to be protected.** The critical components in your circuit that need to be protected should be identified before the product is actually constructed, as it is extremely difficult (if not impossible) to implement proper security mechanisms after-the-fact. Such components may include specific algorithms, device identifiers, digital media, biometrics, cryptographic keys, complete product firmware, or other product-specific data. In addition to protecting discrete data contents, you may be interested in implementing a secure product boot sequence, field programmability, or remote management interface. Be aware that in some cases, even non-critical portions can unknowingly compromise the security of the system.

- **Why it is being protected.** In some countries, protecting certain content may be a legislative requirement (for example, a medical device containing confidential patient information must be secured in order to meet the U.S. HIPAA requirements). In most situations, critical data is being protected to prevent a specific security threat (see Section 3 for details). It is important to acknowledge that an attack threat may exist and to implement the proper mechanisms to prevent them. Ignoring or overlooking the possibility of attack can lead to a vulnerable product.

- **Who you are protecting against.** The types of attackers vary greatly (see Section 3.1 for details), from a curious hardware hacker to an entire group backed by organized crime, government, or a competitor. As such, it is important to attempt to properly identify the skill level and theoretical goals of the primary attackers.

A designer has a challenging task to create and ensure security of their system. Not only are they responsible for the entire system, they must understand every possible aspect of the design and are typically constrained by technical, financial, or political agendas. An attacker has an easier job to achieve their goal, which is to exploit any insecurity of the system. They need to only discover one exploitable area of the design and typically have fewer constraints on their methods than the designer. Attackers invest resources, in the form of time and money, in pursuit of a specific goal. They will likely choose the attack that yields the best results in the easiest and most repeatable fashion. It is extremely important to note that a security mechanism implemented in one product may not be suitable or useful in another.

3. ATTACK AND THREAT CLASSIFICATIONS

The key to understanding how to design a properly protected system is to understand the mindset of an attacker. This section will explore classifications of attackers and look at the most common threat models. Today, with the advances in technology, lower cost of products, and easier access by the public to the once-specialized tools, attacks on hardware are becoming more prevalent.

Attacks generally fall into one of three categories, though some attacks may consist of more than one:

- **Insider Attack.** Insider threats make up a significant portion of computer security breaches. This may come in form of run-on fraud by a manufacturer (producing additional identical units of a product to be sold on the grey market) or a disgruntled employee (willing to sabotage or sell critical product information, such as encryption keys, firmware, or other intellectual property). Many, but not all, insider threats can be thwarted with strict compartmentalization of critical data, access control, and chain of custody policies.

- **Lunchtime Attack.** These attacks often take place during a small window of opportunity, such as a lunch or coffee break. Typically, the attack would need to be completed in a short period of time (ranging from a few minutes to a few hours). Lunchtime attacks are risky, as they can easily be detected if the target product is missing or has visibly been tampered with.

- **Focused Attack.** Time, money, and other resources typically are not an issue for a focused attack, in which the adversary can bring the product into a private location to analyze and attack with no risk of being discovered.

3.1. Attacker Classification

In Abraham, et al.'s *Transaction Security System* [1], attackers are classified into three groups (now essentially an industry-standard) depending on their expected abilities and strengths:

- **Class I (clever outsiders).** They are often very intelligent but may have insufficient knowledge of the system. They may have access to only moderately sophisticated equipment. They often try to take advantage of an existing weakness in the system, rather than try to create one.

- **Class II (knowledgeable insiders).** They have substantial specialized technical education and experience. They have varying degrees of understanding of parts of the system but potential access to most of it. They often have highly sophisticated tools and instruments for analysis.

- **Class III (funded organizations).** They are able to assemble teams of specialists with related and complementary skills backed by great funding resources. They are capable of in-depth analysis of the system, designing sophisticated attacks, and using the most advanced analysis tools. They may use Class II adversaries as part of the attack team.

Table 1 compares each attack classification against available resources, based on Kocher's *Crypto Due Diligence* [23]. The majority of practical design examples presented in this paper are targeted towards thwarting Class I and a portion of Class II attackers.

Table 1. Comparison of Attack Classification with Available Resources.

Resource	**Class I (Curious hardware hacker)**	**Class II (Academic)**	**Class III (Organized crime)**	**Class III (Government)**
Time	Limited	Moderate	Large	Large
Budget ($)	< $1000	$10k - $100k	> $100k	Unknown
Creativity	Varies	High	Varies	Varies
Detectability	High	High	Low	Low
Target/Goal	Challenge/Prestige	Publicity	Money	Varies
Number	Many	Moderate	Few	Unknown
Organized?	No	No	Yes	Yes
Release information about attack?	Yes	Yes	Varies	No

3.2. Attack Difficulty

To classify the difficulty of attack against a product, Weinhart, White, and Arnold specified six levels in *An Evaluation System for the Physical Security of Computing Systems* [43], shown in Table 2. If implemented properly, the majority of practical design examples presented in this paper will give your product up to Level 4 protection.

Table 2. Attack Difficulty Classification.

Level	Name	Description
1	None	The attack can succeed "by accident," without the attacker necessarily being aware that a defense was intended to exist. No tools or skills are needed.
2	Intent	The attacker must have a clear intent in order to succeed. Universally available tools (e.g., screwdriver, hobby knife) and minimal skills may be used.
3	Common Tools	Commonly available tools and skills may be used (e.g., those tools available from retail department or computer stores, such as a soldering iron or security driver bit set).
4	Unusual Tools	Uncommon tools and skills may be used, but they must be available to a substantial population (e.g., multimeter, oscilloscope, logic analyzer, hardware debugging skills, electronic design and construction skills.) Typical engineers will have access to these tools and skills.
5	Special Tools	Highly specialized tools and expertise may be used, as might be found in the laboratories of universities, private companies, or governmental facilities. The attack requires a significant expenditure of time and effort.
6	In Laboratory	A successful attack would require a major expenditure of time and effort on the part of a number of highly qualified experts, and the resources available only in a few facilities in the world.

3.3. Product Accessibility

There are four scenarios in which the attacker can gain access to a product and often corresponds directly to an attack goal (see Section 3.4 for details):

- **Purchase.** The attacker can purchase the product through a retail outlet, often with no means of detection (e.g., paying cash). Multiple products could be purchased, with the first few serving as disposable units to aid in reverse engineering or to discover any existing tamper mechanisms. This scenario may be financially prohibitive for Class I attackers but is typical for most focused attacks.

- **Evaluation.** The attacker can rent or lease the product from the vendor or distributor, often on a monthly basis. Most attacks are possible, though there is a risk of detection since it will need to be returned.

- **Active.** The attacker does not own the target product. The product is in active operation and may belong to a specific person. This is the most difficult type of attack, as risk of detection is quite high. The attacker may still have physical access to the product, if only for a short period of time (e.g., a lunchtime attack), so most attacks are still possible.

- **Remote Access.** The attacker will not have physical access to the product and all attacks will be done remotely (e.g., through a wired or wireless network). The attacker does not require special hardware tools and can easily mask their location. Risk of detection is low.

3.4. Threat Vectors

At the highest level, four classes of security threat exist, as based on Pfleeger's *Security in Computing* [35]:

- **Interception (or Eavesdropping).** Gaining access to protected information without opening the product. On a computer network, this could be done by illicitly copying data or through promiscuous mode network monitoring. With embedded systems, this could be achieved by monitoring the external interfaces of the device or by analyzing compromising signals in electromagnetic radiation, power supply current fluctuations, or protocol timings (see Section 4.4 for details). Although a loss may be discovered fairly quickly (e.g., in the case of credit card theft or spoofed user authentication), a silent interceptor may leave no traces by which the interception can be readily detected.

- **Interruption (or Fault Generation).** An asset of a product becomes lost, unavailable, or unusable. An example is a Denial-of-Service attack, malicious destruction of a hardware device, or intentional erasure of program or data contents. Fault generation falls into this class, which consists of operating the device under abnormal environmental conditions to intentionally provoke malfunctions, which may lead to the bypassing of certain security measures.

- **Modification.** Tampering with an asset of a product. Modification is typically an invasive technique for both hardware (circuit modifications or microprobing, see Section 4.2 for details) and software (changing the values of data or altering a program so that it performs a different computation). Some cases of modification can be detected with simple security measures, but other more subtle changes may be almost impossible to detect.

- **Fabrication.** Creating counterfeit objects on a computing system or product. Fabrication can come in many forms, including a man-in-the-middle attack, inserting spurious transactions into a network, or adding data into a device. Sometimes these additions can be detected as forgeries, but if skillfully done, they may be indistinguishable from the real thing.

Within these classes, specific threat vectors and attack goals exist, some examples being:

- **Competition (or Cloning).** In this scenario, an attacker (usually in the form of a competitor) would reverse engineer or copy specific intellectual property from the product with a primary goal of gaining an advantage in the marketplace (by improving their product using the stolen technology or by selling lower-priced knock-offs).

- **Theft-of-Service.** Generally, these attacks aim to obtain a service for free that usually requires payment. Examples include mobile phone cloning or bypassing copy protection schemes.

- **User Authentication (or Spoofing).** These attacks are typically focused on products that are used to verify the owner's identity, such as a smartcard, biometric reader, or one-time-password generator.

- **Privilege Escalation (or Feature Unlocking).** These attacks are aimed at unlocking hidden/undocumented features of a product or to increase the amount of control given to the user without having legitimate credentials (e.g., acquiring administrator access on a network appliance).

Human nature can be a significant dent in the secure hardware armor. Skilled attackers can use social engineering techniques to obtain sensitive or company confidential information about a specific product or mechanism without ever tampering with the actual device, as discussed in Mitnick and Simon's *The Art of Deception: Controlling the Human Element of Security* [32]. Searching patents, data sheets, and available information on the Web and in libraries can also provide product details that may give an attacker an unintended advantage.

4. PRACTICAL DESIGN SOLUTIONS

This section examines three levels of the product: enclosure, circuit board, and firmware. Design solutions and protection methods are proposed, and attack examples are provided for historical reference.

4.1. Product Enclosure

The design of a secure product enclosure is critical to prevent attackers from gaining access to the internal circuitry. Once the circuit board is accessible to the attacker, they will typically reverse-engineer the design to create a schematic and then identify possible attack vectors. Opening some products is as simple as loosening a few screws or prying open the side with a hobby knife or screwdriver (as shown in Figure 1). Section 4.1.2 looks at tamper mechanisms and other solutions to create a more secure enclosure.

Additionally, consideration at this stage must be given to how the device communicates with the outside world and what information can be retrieved from the device without gaining physical access.

Figure 1. Opening the housing of an Apple iPod using a jeweler's screwdriver, from Grand, et al.'s *Hardware Hacking: Have Fun While Voiding Your Warranty* [14]

4.1.1. External Interfaces

External interfaces are typically a product's lifeline to the outside world. Such interfaces may be used for a number of purposes, including connecting to peripherals, field programming, or testing during product manufacturing. Typical interfaces include Firewire[1], USB[2], RS232, Ethernet, or JTAG IEEE 1149.1[3]. Products often implement development or programming interfaces that are not meant for everyday consumer use, but can benefit an attacker immensely. Simply obfuscating these interfaces with proprietary connector types or hidden access doors or holes is not suitable as they will easily be discovered.

When an attacker gains access to an external interface, they will typically first probe the connections to determine their functionality (if it is non-obvious). This is achieved by monitoring the test points for any device-generated signals (using

1 http://www.1394ta.org
2 http://www.usb.org
3 http://www.ti.com/sc/docs/jtag/jtaghome.htm

a multimeter, oscilloscope, or logic analyzer) and then manually toggling the state of the pins to induce a device response. Knowing the state of the pins can help an attacker make an educated guess on the type of interface the product is using.

Once the interface is known, it is trivial for an attacker to monitor the communications using a dedicated protocol analyzer (e.g., CATC[4]) or software-based tool, such as SnoopyPro[5] for USB, SysInternals' PortMon[6] for serial (RS232) and parallel port, and Ethereal[7] for network protocols. One attack against a known protocol is to generate malformed or intentionally bad packets (using the traffic generation features of a protocol analyzer, for example) and observe the results. If the product does not properly handle errors or illegal packets, a failure may trigger an unintended operation that is useful to the attacker.

Figure 2 shows an example of a proprietary external interface on a hardware authentication key fob. The test points (the five horizontal metal dots) are accessible by simply removing a small plastic sticker from the back of the device housing. The sticker can be replaced after attack, leaving no signs of tampering.

Figure 2. External interfaces on a hardware authentication device

xda-developers.com discovered an attack against an XDA device through its JTAG interface [46]. Although the XDA does not have an external interface specifically used in the attack, the unit simply had to be unscrewed and wires attached to the proper test points. The JTAG functionality was still enabled on the board and was used to read and write the internal Flash ROM.

Kingpin and Mudge's *Security Analysis of the Palm Operating System and its Weaknesses Against Malicious Code Threats* [22] details an attack against devices running the Palm Operating System, which transmits an obfuscated version of the system password over the serial port during a HotSync operation. The encoded password can be retrieved and decoded into its original ASCII form. Additionally, designed into early versions of Palm OS is an RS232-based "Palm Debugger", which provides source- and assembly-level debugging of Palm OS executables. This debugger can be accessed through the HotSync port using commercial tools and enable any user to view raw memory, reset the device, export specific databases, or install, delete, or execute applications.

Use caution when connecting to the "outside world". If possible, encrypt or at least obfuscate traffic to increase attack difficulty. No secret or critical components should be able to be accessed through the external interface. Only publicly known information should be passed.

[4] http://www.catc.com
[5] http://sourceforge.net/projects/usbsnoop
[6] http://www.sysinternals.com/ntw2k/freeware/portmon.shtml
[7] http://www.ethereal.com

Removing external programming or test interfaces may increase complexity of manufacturing or field upgradeability at the expense of security. JTAG functionality should be removed from operational modes if at all possible. If an interface is simply disconnected from the device (by blowing fuses or cutting traces on the PCB during manufacturing, for example), an adversary could easily reconnect or repair the interface and launch an attack.

While physical connections are the focus of this section, it is important to note that wireless interfaces also need to be secured, though beyond the scope of this paper. Arbaugh's Wireless Research Web site [5] details a number of vulnerabilities with 802.11b's[8] inherent encryption and Whitehouse's *War Nibbling: Bluetooth Insecurity* [45] demonstrate that Bluetooth[9] technologies are also at risk.

4.1.2. Tamper Mechanisms

The goal of tamper mechanisms is to prevent any attempt by an attacker to perform an unauthorized physical or electronic action against the device. Tamper mechanisms are divided into four groups: resistance, evidence, detection, and response. Tamper mechanisms are most effectively used in layers to prevent access to any critical components. They are the primary facet of physical security for embedded systems and must be properly implemented in order to be successful. From the designer's perspective, the costs of a successful attack should outweigh the potential rewards.

Often, existing tamper mechanisms can only be discovered by attempted or complete disassembly of the target product. This may require an attacker to obtain more than one device in order to sacrifice one for the sole purpose of discovering such mechanisms. Once the mechanisms are noted, an adversary can form hypotheses about how to attack and bypass them.

Weingart's *Physical Security Devices for Computer Subsystems: A Survey of Attacks and Defenses* [44] describes physical tamper mechanism attacks and defenses ranging from cheap and easy to extremely costly and complex. It is a comprehensive guide to many (if not all) known attack types and provides list of solutions to implement to protect against such attacks. Physical attacks include different types of probing (passive, active/injector, pico-probes, or energy), machining methods (manual material removal, mechanical, water, laser, chemical, or shaped charge), and electrical (radiation imprinting, temperature imprinting, high voltage imprinting, power supply fluctuations, clock glitching, circuit disruption, or electron beam and infrared laser read/write). Corresponding countermeasures for these attacks are described, including various types of physical barriers (hard barriers, single chip coatings, or insulator-based substrates), tamper evident solutions (brittle packages, crazed aluminum, polished packages, bleeding paint, or holographic tape), tamper detection sensors (voltage, probe, wire, printed circuit board, flex, stressed glass, piezo-electric, motion, ultrasonic, microwave, infrared, acceleration, radiation, flux, dosage, or temperature), and tamper response technologies (RAM power drop, RAM overwrite, or physical destruction).

Clark's *Physical Protection of Cryptographic Devices* [9] is a survey of attack risks, objectives, and scenarios related to tamper mechanisms and secure hardware design.

4.1.2.1. Tamper Resistance

Tamper resistance consists of using specialized materials to make tampering of a device or module difficult. This can include such features as hardened steel enclosures, locks, encapsulation, or security screws. Implementing tight airflow channels (that is, tightly packing the components and circuit boards within the enclosure) will increase the difficulty of optical probing of the product internals using fiber optics. A side benefit of many tamper resistant mechanisms is that they are often tamper evident, meaning that physical changes can be visually observed and it becomes obvious that the product has been tampered with (see Section 4.1.2.2 for details).

[8] http://standards.ieee.org/getieee802
[9] http://www.bluetooth.com

If designing a housing that requires screws, or when retrofitting a design that is already using screws, consider implementing one-way screws that will offer additional tamper resistance. Although an adversary can likely drill through such screws, they raise the difficulty of attack over an industry-standard screwdriver or Torx driver bit. The Thomas Register Directory provides a large listing of security- and tamperproof-screw manufacturers and suppliers[10].

Sealing both sides of the housing together in such a way that requires the destruction of the device in order to open it should be considered. Many plastics are sensitive to heat and melt at fairly low temperatures. Consider sealing the housing with high-temperature glue or ultrasonic welding to reduce tampering. If using high-temperature glue, choose one with a higher softening point than the plastic housing in order to increase visible tamper evidence. Serviceability may be an issue if the product is intended to be opened by authorized personnel. However, if a legitimate user can open the device, an adversary can, too.

Encapsulating the entire circuit board with resistant resin or epoxy compound will help to protect the circuitry. However, it is more common for such encapsulation to be done on only specific critical components. Conformal coatings and encapsulants are typically used to protect an assembled circuit board from moisture, fungus, dust, corrosion, or tampering. It can also reduce mechanical stress on components and protect them from thermal shock. Urethane provides a hard, durable coating that offers excellent abrasion and solvent resistance. It shrinks significantly during coating, which may stress components. Epoxies also offer excellent resistance to moisture and solvents. Usually consisting of a two-part thermosetting resin, the coating also shrinks during curing, leaving a hard, difficult to remove film. Conformal coatings are provided by a large number of manufacturers, including 3M, GE Silicones[11], Dow Corning[12], and MG Chemicals[13].

Chemicals exist, such as methylene chloride, sulfuric acid, fuming nitric acid, or MG Chemicals' Conformal Coating Stripper[14], which may remove protective coatings, so be sure to evaluate that your chosen compound is suitable for your desired protection level. In order to protect against a chemical attack that removes the encapsulation, aluminum powder can be added to the compound. A solvent capable of dissolving the aluminum will corrode the underlying components or circuitry, rendering the device useless.

4.1.2.2. Tamper Evidence

The goal of tamper evidence is to ensure that there is visible evidence left behind when tampering occurs. Tamper evident mechanisms are a major deterrent for minimal risk takers (e.g., non-determined attackers). There are hundreds of tamper evident materials and devices available, mostly consisting of special seals and tapes to make it obvious that there has been physical tampering.

Tamper evidence features are only successful if there is a process in place to check if tampering has occurred or if a legitimate owner of the device notices a deformity. Generally speaking, if an adversary purchases a product with the specific intention of attacking it, tamper evident mechanisms by themselves will not prevent the attack.

In Johnston and Garcia's *Vulnerability Assessment of Security Seals* [19], the authors explain 94 different security seals (including adhesive tape, plastic, wire loop, metal cable, metal ribbon, bolt type, secure container, passive fiber optic, and electronic) were defeated using low-cost tools and readily availably supplies. The time for a successful attacked ranged from 3 seconds to 125 minutes and the cost of necessary attack tools ranged from $0.15 to $750. Weingart's *Physical Security Devices for Computer Subsystems: A Survey of Attacks and Defenses* [44] provides dozens of potential tamper evident mechanisms to employ. Most (if not all) of the available tamper evident seals can be bypassed.

[10] http://www.thomasregisterdirectory.com/screws/tamperproof_screws_0012296_1.html
[11] http://www.gesilicones.com
[12] http://www.dowcorning.com/content/etronics
[13] http://www.mgchemicals.com
[14] http://www.mgchemicals.com/products/8310.html

Holdtite[15] manufactures Secure 42, superglue intended to provide evidence of tampering. Brittle plastics or enclosures that crack or shatter upon an attempted penetration may be suitable in certain environments. "Bleeding" paint, where paint of one color is mixed with tiny spheres of a contrasting color paint which rupture when the surface is scratched is a novel solution.

4.1.2.3. Tamper Detection

Tamper detection mechanisms enable the hardware device to be aware of tampering and typically fall into one of three groups:

- **Switches** such as microswitches, magnetic switches, mercury switches, and pressure contacts to detect the opening of a device, the breach of a physical security boundary, or the movement of a particular component.

- **Sensors** such as temperature and radiation sensors to detect environmental changes, voltage and power sensors to detect glitch attacks, radiation sensors for X-rays (used for seeing what is inside of a sealed or encapsulated device) and ion beams (often used for advanced attacks to focus on specific electrical gates within an integrated circuit).

- **Circuitry** such as flexible circuitry, nichrome wire, and fiber optics wrapped around critical circuitry or specific components on the board. These materials are used to detect if there has been a puncture, break, or attempted modification of the wrapper. For example, if the resistance of the nichrome wire changes or the light power traveling through the optical cable decreases, the system can assume there has been physical tampering.

Again, Weingart's *Physical Security Devices for Computer Subsystems: A Survey of Attacks and Defenses* [44] provides a comprehensive list of specific mechanisms that could be employed.

4.1.2.4. Tamper Response

Tamper response mechanisms are the countermeasures taken upon the detection of tampering. Chaum's 1983 *Design Concepts for Tamper Responding Systems* [8] presents concepts for implementing sensors into tamper responsive systems.

Most often, the response consists of completely shutting down or disabling the device, or erasing critical portions of memory to prevent an attacker from accessing secret data. Physical destruction of a device using a small explosive charge may be an option for extremely secure devices, but is not practical for most (if any) consumer electronics. Response mechanisms may also be simpler, such as just log the type of attack detected and the time it occurred, which can provide useful audit information and help with forensic analysis after an attack.

Simply erasing critical portions of memory (also known as "zeroizing") is usually not enough, however, as shown by Gutmann's *Secure Deletion from Magnetic and Solid-State Memory Devices* [16] and *Data Remanence in Semiconductor Devices* [15] and Skorobogatov's *Low Temperature Data Remanence in Static RAM* [39]. Gutmann observes that "contrary to conventional wisdom, volatile semiconductor memory does not entirely lose its contents when power is removed. Both static (SRAM) and dynamic (DRAM) memory retains some information." Section 4.2.3 explores this issue in more detail.

W.L. Gore's D3 electronic security enclosures[16] are designed to protect the physical security boundary of a module and combine a number of tamper evidence and detection features. The sensor comes as a foldable sheet that is to be wrapped around the product. Conductive ink crisscrosses through the sheet with a maximum distance between traces of 200 to 300 microns (a pitch too small to be drilled through without detection). The electrical state of the sensor changes if the

[15] http://www.holdtite.com/english/handbook/spectronix/tproof.htm
[16] http://www.goreelectronics.com/products/specialty/Electronic_Security_Enclosures.html

field is broken, which will trigger the product to enable its tamper respondent mechanisms. Gore claims that the device is transparent to X-rays (which may be used to determine the location of the sensor within the product) and that it has been tested against a wide range of reagents and solvents. The outer layer has an opaque resin coating, which conceals all surface details of the sensor and prevents an attacker from seeing any traces. This product also meets the requirements of FIPS 140 Level 4 Specification for Cryptographic Modules [34].

It is unlikely that there will be accidental triggers of tamper response mechanisms. Even still, the legitimate user will need to understand the environmental and operational conditions and keep the device within those limits. Many tamper-responsive devices are designed and manufactured with the stipulation that they will never be opened - legitimately or not.

4.1.3. Emissions and Immunity

The prevention of "compromising emissions" is an important requirement for secure hardware. All electronic devices generate electromagnetic inference (EMI) in one form or another. Emission security can be traced back as early as World War I, when field telephone lines could be monitored using the phenomenon of "cross talk," interference caused by the energy from one telephone conversation invading another by electrostatic or electromagnetic coupling [4]. Due to the sensitive nature of the subject, most research was classified and it soon disappeared from public literature. Much of the government shielding requirements (known as "TEMPEST") remains to be classified by the U.S. Government. However, many unclassified documents are available on John Young's TEMPEST Documents Web site [48].

van Eck's 1985 *Electromagnetic Radiation from Video Display Units: An Eavesdropping Risk?* [41] was the first academic article on the subject, which described the results of research into eavesdropping on video display units by picking up and decoding the electromagnetic interference. More recently, Kuhn and Anderson's *Soft Tempest: Hidden Data Transmission Using Electromagnetic Emanations* [29] showed how compromising emissions from a PC could be made better or worse by using specific font styles and colors. Loughry and Umphress's *Information Leakage from Optical Emanations* [30] describes how light emitting diodes (LEDs) in certain communications equipment leak the data contents that are being transmitted, and details a number of design changes to reduce such risk.

Kocher, Jaffe, and Jun's *Differential Power Analysis* [26], [27] (DPA) describes the monitoring of the electrical activity of a smart card and using mathematical methods to determine secret information stored in the device (such as user PINs or cryptographic keys). Simple Power Analysis (SPA) is a predecessor to DPA in which an attacker directly observes a system's power consumption, which varies based on the operation that the microprocessor is performing. Intensive operations, such as cryptographic functions, can easily be identified. While SPA attacks primarily use visual inspection to identify relevant power fluctuations, DPA attacks use statistical analysis and error correction techniques to extract information correlated to secret keys.

Rao and Rohatgi's *EMPowering Side-Channel Attacks* [37] provides preliminary results of similar attacks by measuring EMI. These types of focused EMI and power attacks are useful on small, possibly portable devices, such as smart cards or authentication tokens. Larger devices, such as desktop computers and network appliances, might generate too much EMI to be able to measure specific, minute changes. Well-filtered power supplies and physical shielding can make attacks infeasible.

Eavesdropping on EMI emissions is referred to as a passive attack. An active attack consists of directing high-energy RF (HERF) signals or directing electrostatic discharge (ESD) at the product in order to cause failures. Properly designing a product to meet specific EMI and RF emissions conditions is part of many specifications, including those of the FCC, EU, IEC, and FDA. Essentially the inverse of emissions testing, immunity testing subjects a product to various RF phenomena to see if it affects the product's performance. ESD protection components are often designed into external connectors and contacts to reduce the chance of failure. One attack uses an ESD simulator tool to generate a high voltage spike and inject it into a device's external interface or keypad in hopes of causing an unexpected or unintended condition (for example, by causing the program counter of the microprocessor to jump to a different code portion or change the values on the address or data bus). However, unless the injection of HERF or ESD can be reproduced in a controlled manner, the results may be too unpredictable to be useful.

At the enclosure level, EMI shielding can easily be designed in or retrofitted to a design in the form of coatings, sprays, tapes, or housings in order to decrease emissions and increase immunity. If the enclosure is metal, EMI tape can be used to seal any slots or seams in the case. If the case is plastic, EMI spray (a conductive paint) can be used to coat the inside of the case. EMI tapes can come lose, causing short circuits, and EMI paint can flake or chip if the surface is not cleaned prior to application. Also, be aware of any changes in thermal characteristics that EMI shielding may cause. EMI shielding solutions are provided by a large number of manufacturers, including W.L. Gore[17], Schlegel Systems[18], and GC/Waldon Electronics[19]. The Thomas Register Directory also provides a large listing of EMI/RFI shielding manufacturers and suppliers[20]. More specific design features at the board level in order to reduce compromising EMI emissions are detailed in Section 4.2.

4.2. Board Level

Many of the weaknesses, security vulnerabilities, and design flaws of a product are identified when analyzing the circuit board. Anderson and Kuhn's *Low Cost Attacks on Tamper Resistant Devices* [3] describes a number of common attack techniques that low-budget adversaries can use to break smart cards and "secure" microcontrollers. There are a number of steps, both basic and advanced, that can be implemented at the circuit board level to help prevent some attacks and mitigate some risk.

Simple attacks range from reading or modifying the contents of a microprocessor or memory device, or replacing components on the board. More advanced attacks involve microprobing, in which a chip package is opened, its internals accessed with semiconductor test equipment, and the internal data paths observed or manipulated, or fault generation attacks in which the device is operated under environmental stress conditions outside its designed operational range (such as extreme temperature, supply voltage variations and spikes, protocol violations, and partial system resets).

4.2.1. Physical Access to Components

Sensitive components that are most likely to be targeted for an attack (such as the microprocessor, ROM, RAM, and programmable logic) should be made difficult to access.

Reverse engineering the target product usually requires one to determine the part numbers and device functionality of the major components on the board. Understanding what the components do may provide details for particular signal lines that may be useful for active probing during operation. Components are easily identified by their part numbers and manufacturing markings on the device packaging [18], and by following their traces to see how they interconnect with other components on the board. Nearly all IC manufacturers post their component data sheets on the Web for public viewing, and online services such as IC Master[21], Data Sheet Locator[22], and PartMiner[23] provide part number searches and pinout and package data for hundreds of thousands of components. To increase the difficulty of reverse engineering and device identification, it is recommended that all markings be scratched off the tops of the chips.

Using BGA packages increases the difficulty of casual probing, manipulation, and attack, due to the fact that all die connections are located underneath the device packaging. However, manufacturing costs are more expensive for BGA compared to other package types due to the fact that X-rays are often used to verify that the solder has properly bonded to each of the ball leads. A dedicated attacker with common circuit board rework equipment could easily remove the

[17] http://www.goreelectronics.com/products/emi/Emi.html
[18] http://www.schlegel.com/EMI_shielding
[19] http://www.gcwaldom.com
[20] http://www.thomasregisterdirectory.com/emi_rfi_shielding/index.html
[21] http://www.icmaster.com
[22] http://www.datasheetlocator.com
[23] http://www.freetradezone.com

device and add a socket for easier access. Because of this, it is recommended to place critical devices in areas of the circuit board that may not have enough area or vertical height around the component for a socket to be properly mounted.

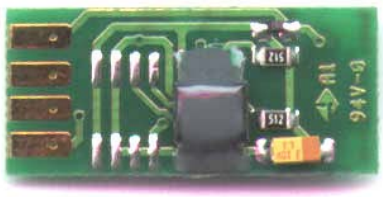
It is also recommended to add some type of epoxy encapsulation or glue to help prevent easy removal of components. Figure 3 shows a Sprint PCS cellular telephone with conformal coating around the edges of three critical BGA devices (microprocessor, Flash, and SRAM). Manufacturing cost to add this to the process should be minimal.

Figure 3: Glue around the edges of BGA packages on Sprint PCS phone

Another solution is to employ Chip-on-Board (COB) packaging, in which the silicon die of the integrated circuit is mounted directly to the PCB and protected by epoxy encapsulation. Even though methods exist to gain access to COB devices (discussed later in this section), and an attacker may still probe vias and traces extending from the encapsulate, direct manipulation with the device and its connections are less of a threat. Using COB devices also increases manufacturing cost and is not necessarily supported by all manufacturers, as specialized equipment is required to manipulate the wire bonds between the silicon die and the circuit board. A relatively new technology known as Chip-in-Board (CIB) embeds the silicon die within the layers of a printed circuit board. The concept is similar to COB, though a cavity is created in the circuit board to hold the die. An encapsulate is filled in over the die and cavity, creating a flat PCB surface. It is unknown what the added financial burden of this technology is. Using CIB in conjunction with buried vias to completely hide critical traces is a possibility (see Section 4.2.2 for more).

When epoxy encapsulate is incorporated into a design to protect components, ensure that it serves its intended purpose. Figure 4 shows an example of an early USB authentication device that stored critical data on the encapsulated Serial EEPROM. Aside from being able to scrape off the epoxy using a hobby knife to gain access to the surface-mount pins of the device, an attacker could simply solder wires to the exposed footprint adjacent to the device, which was intended for another Serial EEPROM, and read the memory contents using an industry-standard device programmer. This misuse of epoxy coupled with the easy accessibility of the device played a major role in the successful attack of the product, as detailed in Kingpin's *Attacks on and Countermeasures for USB Hardware Token Devices* [20]. Additionally, by repeatedly changing the user password of the device and reading the EEPROM after each change, it was possible to determine if the password was being stored in memory, where it was being stored, and what type of obfuscation, encoding, or encryption (if any) was used.

Figure 4: Early Rainbow iKey showing epoxy encapsulation that still allows an attacker to access the adjacent footprint



4.2.2. PCB Design and Routing

Proper engineering practices should always be exercised. Traces should remain as short as possible. Differential signal lines should be aligned parallel even if located on separate layers. Noisy power supply lines should be kept away from sensitive digital and analog lines. Properly designed power and ground planes should be employed to reduce EMI emissions. Additionally, any unnecessary test points should be removed from the design, as they allow unwanted noise and interference to pass through the PCB. If testpoints are required, consider using a copper-filled pad as opposed to a through-hole pad.

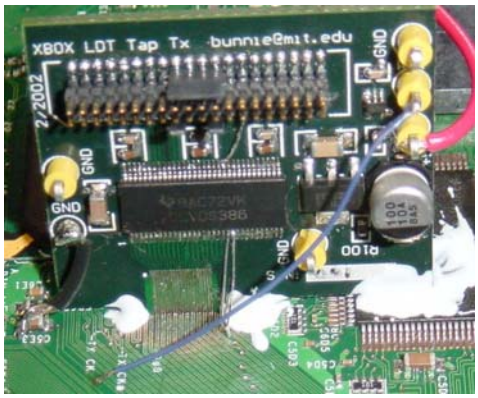
Critical traces should be hidden on inner board layers and trace paths should be obfuscated to prevent easy reverse engineering of circuitry. Use buried vias, which connect two or more inner layers but no outer layer and cannot be seen from either side of the board, to reduce potential probing points for the attacker. Be aware of electrical noise issues that these modified traces may introduce.

4.2.2.1. Bus Protection

Device operation and information can be gleaned by analyzing the internal address, data, and control bus lines with a logic analyzer, digital oscilloscope, or custom circuitry. Targeted bus lines could be probed by simply removing the soldermask on the circuit board. Be aware of the data being stored in memory at all times and what is transferred across exposed and accessible buses.

Huang's *Hacking the Xbox: An Introduction to Reverse Engineering* [17] details the creation of a tap board used to intercept data transfer over the Xbox's HyperTransport bus (see Figure 5). Huang was able to retrieve the symmetric encryption key used for protection of a secret boot loader, which ultimately allowed him to executed untrusted code on the system. Moving critical bus lines onto internal layers would thwart such an attack. If a multi-layer board is not used, protective encapsulant could be applied to the target traces.

Figure 5: Huang's Xbox HyperTransport Bus tap circuit, from http://www.xenatera.com/bunnie/proj/anatak/xboxmod.html



4.2.3. Memory Devices

Most memory devices are notoriously insecure. Some memory devices employ security features to prevent regular device programmers from reading stored data, such as physical fuses on ROMs and boot-block protection in Flash. Reading RAM or other volatile storage areas while the device is in operation may yield temporarily stored data or plaintext components. The Dallas Semiconductor DS2432 EEPROM[24] is an example of a secure memory device that uses the Secure Hash Algorithm (SHA-1) and a user-provided write-only secret to protect stored data. The Atmel CryptoMemory family of devices[25] includes EEPROMs and synchronous and asynchronous Flash with authentication, password, and encryption features. Most standard memory devices do not have this type of functionality and are readable, often in-circuit, with standard tools. For those devices that have boot block locking or security bit features, be sure to use them to sufficiently raise the bar against Class I attackers who may be trying to clone or reverse engineer the device.

IC delidding, for the purpose of gaining access to the silicon die of the IC, is difficult to perform without the use of proper tools because hazardous chemicals are often required and the underlying die is very fragile. However, many academic institutions have access to such equipment. Decapsulation products are also offered by a handful of companies specializing in failure analysis, including Nippon Scientific[26] and ULTRA TEC Manufacturing[27]. So, although the attacks are advanced, the tools required to perform them are available with proper funding. Additionally, reverse engineering services are offered by companies such as Semiconductor Insights[28] that aid in functional investigation, extraction, and simulation of ICs. They can also analyze semiconductor and fabrication processes, techniques, and materials.

With access to the die, one can bypass many of the available security mechanisms and may be able to determine the contents of the device. Beck's *Integrated Circuit Analysis - A Guide to Preparation Techniques* [6] and Kömmerling and Kuhn's *Design Principles for Tamper-Resistant Smartcard Processors* [28] provide details of such techniques. It is difficult to prevent against reading ROM devices, since each cell is a physical broken or intact connection. However, it is possible to add a top-layer sensor mesh as described in [28] that may disrupt microprobing techniques of an active (e.g., while the device is functioning) attack. This will not prevent an attacker from performing microprobing on an inoperable device (e.g., one that has had power removed).

4.2.3.1. PLDs and FPGAs

Depending on the product, protecting your intellectual property inside of programmable logic devices (PLDs) and field programmable gate arrays (FPGAs) can be just as important as protecting firmware and data in memory. Dipert's *Cunning Circuits Confound Crooks* [10] provides a good look at the security-related advantages and disadvantages of various PLD and FPGA technologies. Essentially, SRAM-based devices are the most vulnerable to attack due to their requirement to have configuration memory external to the device (stored in separate nonvolatile memory or program firmware), which is then loaded into the FPGA upon power-up. The bit stream between the configuration memory and FPGA simply needs to be monitored to retrieve the entire FPGA configuration.

Be sure to implement protection against simple I/O scan attacks, in which an adversary attempts to reverse engineer a programmable logic design by cycling through all possible combinations of inputs and then monitoring the outputs to determine the internal logic functions. This type of attack is easiest against low-density PLDs with dedicated inputs and outputs and for designs containing only asynchronous circuits and latches. A solution would be to use unused pins on the device to detect probing or tampering. Pins could be set to inputs, and if they detect a level change, the device can assume it is being probed and perform a countermeasure or response. Additionally, when designing state machines in

[24] http://pdfserv.maxim-ic.com/en/ds/DS2432.pdf
[25] http://www.atmel.com/products/securemem
[26] http://www.nscnet.co.jp/e
[27] http://www.ultratecusa.com
[28] http://www.semiconductor.com

FPGAs, ensure that all conditions are covered and there are defaults in place for unused conditions. Otherwise, an attacker may be able to put the FPGA into an indeterminate state through fault generation attacks.

Consider adding digital "watermarks" to your design in the form of unique features or attributes that can be used later, if necessary, to prove that a design claimed by a competitor to be original is actually a copy. With any type of device used, take advantage of any on-chip security features that the device may offer. Even enabling the simple security fuse/software protection bit adds a level of protection compared to not enabling it at all.

Some product types worth investigating further are Actel's Antifuse FPGAs[29] and QuickLogic FPGAs[30], both of which eliminate the need for external configuration memories required by SRAM-based FPGAs.

4.2.3.2. Advanced Memory Management

Advanced memory management consists of using an FPGA or other circuitry to perform hardware-based bounds checking by monitoring the address bus or buses. By doing so, one can restrict read/write access to defined memory locations and could trigger a tamper response mechanism if address access is outside of the defined range. Such technology is commonly implemented in secure coprocessors (see Section 4.2.7).

4.2.4. Power Supply

Precautions should be taken to prevent intentional variation of the power and clock. Minimum and maximum operating limits should be defined and protected using comparators, watchdogs, or supervisory circuitry (available from numerous manufacturers including Maxim[31] and Linear Technology[32]). Do not rely on the end user to supply a voltage within the recommended operating conditions. Using a low-dropout linear regulator or DC-DC converter will help ensure that the circuitry in the product receives power within its expected range, regardless of an improper voltage supplied at the input. Such circuitry can obviously be bypassed if the attacker has access to the board.

To aid in the reduction of EMI, noisy circuitry (such as power supply components) should be compartmentalized to one area of the board and supported with proper filtering. Additionally, power supply circuitry should be physically as close to the power input as possible.

An example of an attack using power supply variation is one with the PIC16C84 microcontroller, in which an adversary can clear the security bit without erasing the remaining memory, giving complete access to the once-protected area. This is achieved by raising VCC to VPP-0.5V (approximately 13.5V) during repeated write access to the security bit [36].

As introduced in Section 4.1.3, SPA and DPA attacks monitor the power consumption or electrical activity of a device in order to determine secret information or cryptographic functionality. Essentially, these attacks work because the amount of power consumed by a microprocessor (and thus the rest of the system) varies based on the operation it is performing. Messerges' *Power Analysis Attack Countermeasures and Their Weaknesses* [31] looks at five countermeasures previously proposed to prevent such attacks and discusses the weaknesses of each, including a noise generator using power randomization, power signal filtering using active and passive filters, detachable power supplies, and time randomization by desynchronizing the clock.

[29] http://www.actel.com
[30] http://www.quicklogic.com
[31] http://www.maxim-ic.com
[32] http://www.linear-tech.com

4.2.5. Clock and Timing

Timing attacks rely on changing or measuring the timing characteristics of the circuit and usually fall into one of two categories:

- **Active timing attacks** are invasive attacks requiring physical access to the clock crystal or other timing circuitry. Classified as a fault generation attack, the main goal is to vary the clock frequency to induce failure or unintended operation. Circuits that make use of the clock crystal for accurate timing, such as a time-based authentication token, could be attacked to "speed up" or "slow down" time based on the clock input. Slowing down a device can also help for debugging and analysis that may not be possible at higher rates. Adding countermeasures to disable the system or enable other tamper response mechanisms if the clock is removed may help thwart attacks. To prevent clock-skewing attacks, a Phase-Locked Loop (PLL) could be implemented to help reduce the clock delay and skew within a device. This will also regulate the internal system timing to compensate for variances in clock crystals.

- **Passive timing attacks** are non-invasive measurements of computation time in order to determine data or device operation. By going with the notion that different computational tasks take different amounts of time, it might become possible to determine secret components or break the cryptographic system of the device under attack. Kocher's *Timing Attacks on Implementations of Diffie-Hellman, RSA, DSS, and Other Systems* [25] is a leading text on this subject.

4.2.6. I/O Port Properties

In order to prevent against ESD attacks (introduced in Section 4.1.3), it is recommended to design ESD protection devices onto any connectors or I/O pins that are exposed (such as keypads, buttons, switches, or displays). ESD protection can simply be in the form of clamping diodes or Transient Voltage Suppressor (TVS) devices. Manufacturers include Vishay[33], Fairchild[34], and Semtech[35].

All unused I/O pins should be disabled or set to a fixed state. For example, the Motorola MC68328 DragonBall processor enables the CLKO (Clock Output) pin by default upon reset. CLKO is used for testing of the internal 16.67MHz PLL and outputs a 16MHz sinewave on the pin. If not disabled during normal device operation, the extraneous signal may cause unwanted noise. As discussed with PLDs and FPGAs (Section 4.2.3.1), unused I/O pins could also be configured to detect probing or tampering by setting them to inputs and waiting for a level change. If one is detected, the device can assume it is being probed and perform a countermeasure or response. This type of detection mechanism would only work while the device is active, which is the most likely time for probing by an attacker to occur.

4.2.7. Cryptographic Processors and Algorithms

The secure coprocessor originated from the notion of a protected subsystem that can execute sensitive functions in a trusted manner. This topic is extremely broad and cannot be covered in sufficient depth in this paper.

Programmable, secure coprocessors typically contain a number of essential ingredients including: hardware tamper response, randomness, layered design, self-initialization, authentication, general-purpose processor and coprocessor, persistent storage, and third-party programming interface. If possible, any cryptographic functions in your design should be moved out of firmware and into a dedicated cryptographic device. Physical security is a central assumption upon

[33] http://www.transzorb.com/diodes/protection-tvs-esd
[34] http://parametric.fairchildsemi.com/ss.asp?FAM=TVS
[35] http://www.semtech.com

which secure distributed systems are built; without a secure foundation, even the best cryptosystem or the most secure kernel will fail.

Yee's *Using Secure Coprocessors* [47] provides a well-rounded look at secure coprocessor design and implementation. The IBM 4758[36] is likely the most recognized, commercially available secure coprocessor. Its design has been presented through a number of academic papers and articles, including Smith and Weingart's *Building a High-Performance, Programmable Secure Coprocessor* [40] and Dyer, et al.'s *Building the IBM 4758 Secure Coprocessor* [11]. Bond and Clayton [7] provide the first known attack against the IBM 4758 by taking advantage of a flaw in the "Common Cryptographic Architecture" (CCA) support software to export any and all of the program's DES and 3DES keys. IBM also has a PCI-X Cryptographic Coprocessor[37]. Other vendors providing cryptographic devices include Hi/fn[38], SafeNet[39], Philips Semiconductor VMS747 Security Processor[40], and Rainbow Technologies' CryptoSwift HSM[41]

The strength of cryptography relies on the secrecy of a key (contained in the user hardware), not the employed algorithm. However, it is not sufficient to assume that a large key size will guarantee security. Even if a trusted encryption algorithm is used, such as DES or AES, improper implementation could make the product easy to break. One must have a complete understanding of the requirements and functionality of an encryption solution before it is implemented into a system. Many times, product companies will claim they use "encryption" in their product when in reality it is nothing more than a simple encoding scheme (usually some type of logic operation). Generating custom encryption solutions, also known as "rolling your own", is typically a bad idea as they end up being very insecure. Schneier's *Applied Cryptography* [38] is a comprehensive reference describing the history of cryptography and presenting dozens of cryptographic protocols, algorithms, and source code listings.

An example of improperly used encryption can be seen in Kingpin's *DS1991 MultiKey iButton Dictionary Attack Vulnerability* [21]. The DS1991 iButton is a hardware authentication token that has three internal 48-byte data areas, each protected by a distinct password. Only the correct password will grant access to the data stored within the area. While the marketing literature claimed that "false passwords written to the DS1991 will automatically invoke a random number generator that replies with false responses...which eliminates attempts to break security by pattern association," it was determined that the data returned on an incorrect password attempt is not random at all and is calculated based on the input password and a constant block of data stored within the DS1991 device.

Kingpin and Mudge's *Security Analysis of the Palm Operating System and its Weaknesses Against Malicious Code Threats* [22] details another "roll your own" encryption scenario in which the Palm system password is XOR'ed against a constant block (common across all Palm devices). The system password can be retrieved and decoded into the original ASCII password, giving the attacker access to previously password-protected data.

4.3. Firmware

Firmware typically contains the program code that controls the underlying hardware of the system. Retrieving and analyzing firmware can allow the attacker to gain a detailed understanding of the product and possibly modify code to bypass failure detection or authentication routines. This section provides some recommendations that can be implemented in firmware to help increase the security of the overall product.

[36] http://www-3.ibm.com/security/cryptocards/html/pcicc.shtml
[37] http://www-3.ibm.com/security/cryptocards/html/pcixcc.shtml
[38] http://www.hifn.com
[39] http://www.safenet-inc.com
[40] http://www.semiconductors.philips.com
[41] http://www.rainbow.com/products/cryptoswift/HSM.asp

4.3.1. Secure Programming Practices

Secure programming practice is essential in any programming environment, whether on a constrained embedded system or a more powerful desktop PC. Buffer overflows [2] are possibly the most familiar and common type of attack against improperly written programs, which can be used to crash the device, execute untrusted code, elevate the privilege of the adversary, or perform unintended functions. Graff and Van Wyk's *Secure Coding: Principles and Practices* [13] and Viega and Messier's *Secure Programming Cookbook for C and C++* [42] provide essential reading on the topic.

When compiling firmware for use in a production device, it is recommended to:

- **Remove unnecessary functionality**, including debug routines.
- **Remove symbol tables and debug-specific information**. Selecting a checkbox in the development environment or providing an additional command-line switch often accomplishes this.
- **Use compiler optimizations** to increase the efficiency of the code and possibly obfuscate easily identifiable code segments.

4.3.2. Storing Secret Components

In *Data Remanence in Semiconductor Devices* [15], Gutmann showed that it is extremely difficult to securely and totally erase data from RAM and non-volatile memory. This means that remnants of temporary data, cryptographic keys, and other secrets may still exist and be retrievable from devices long after power has been removed or after the memory contents have been rewritten. Because of this, the current best practice is to limit the amount of time that critical data is stored in the same regions of memory. Storing data in a fixed RAM location can lead to "burn in" and other phenomena that will enable data recoverability even if power is removed from the volatile device. Either move the secret around to different RAM locations (while overwriting the previous area) or periodically flip the stored bits of the secret as described by Gutmann. Furthermore, experiments in Skorobogatov's *Low Temperature Data Remanence in Static RAM* [39] show that temperature plays a role in the retention of data, but that retention time is unpredictable even between two of the same memory device.

Anderson and Kuhn's *Low Cost Attacks on Tamper Resistant Devices* [3] gives mention to a case where the master key for a security module commonly used in banks to generate and check the personal identification numbers (PINs) with which customers authenticate themselves at ATMs, was "burnt" into the SRAM and found to be 90% intact when retrieved.

4.3.3. Run Time Diagnostics and Failure Modes

Run-time diagnostics should be designed into the system to ensure that the device is fully operational at all times. This could consist of enabling internal watchdog functionality and performing periodic system checks. Checksums or hashes of critical memory areas (both ROM and RAM) could be calculated to make sure no data has been changed outside of expected operation.

It is also important to know how your system will respond to failures, either in a "fail open" or "fail closed" fashion. Most systems should fail closed in the event of a failure detection, meaning that the device will be halted or shutdown until the fault is remedied. Strict specifications for failure modes exist depending on the type of product being designed. This is especially true for life-critical medical systems governed by the FDA.

4.3.4. Field Programmability

The proliferation of field-upgradeable hardware has given adversaries an opportunity to attack a product even when not in possession of it. Many vendors provide updated firmware for their products on public facing Web sites. An attacker could easily disassemble and analyze the code with no risk of detection. The attacker may then be able to reverse engineer the system operation details or extract critical information from the firmware image.

Some vendors choose to compress the image as a form of obfuscation. However, it is usually trivial to determine the compression scheme based on known identifiers of common compression routines. Encryption is a much better solution for secure firmware distribution. In addition, using digital signatures or hashes will verify that the firmware image has not been tampered with after leaving the vendor.

4.3.5. Obfuscation

Although "security through obscurity" does not work in the long term, especially once the obscurity is discovered. It is usually better than no protection at all, however, and will be sure to discourage the most basic Class I attackers by making reverse engineering more difficult (at least temporarily). One extremely important note is that employing the methods discussed in this section may provide a false sense of security unless they are coupled with layers of true secure hardware mechanisms.

Fisher's *Protecting Binary Executables* [12] is a decent starting point for implementing obscurity functionality into firmware and discusses encoding string data, using a custom operating system, scrambling address lines through extra logic, replacing library functions, and writing lousy code that may be difficult to reverse engineer. Other obfuscation ideas involve encoding data stored in ROM or RAM by changing the least-significant-bit (LSB) and most-significant-bit (MSB) order in every other location, or adding spurious and meaningless data on unused pins or interfaces (known as "signal decoys"). Such types of obfuscation will not take a determine attacker very long to uncover, at which point the measures are moot.

5. CONCLUSION

The goal of this paper was to introduce the reader to secure hardware design concepts, attacks, and potential solutions. It is by no means complete as such mechanisms are constantly evolving. The beginning sections of the paper provided information on security policies and a baseline classification of attackers, attack types, and threat vectors. The majority of the paper focused on the many aspects of the secure hardware design process, divided into enclosure, circuit board, and firmware layers. Wherever possible, we referenced previous hardware attacks for educational and historical purposes.

When designing a product, it is essential to first establish a security policy that defines the security goals of the product, as you must first understand what you are protecting and why you are protecting it before security can be successfully implemented. Staying aware of the latest attack methodologies and trends will enable you to choose the proper means of protection for your particular product.

It has been said that the only way to stop a hacker is to think like one. Try to break the security of your product. Fix it, and try to break it again. Allow time for this iterative process during the design cycle. Do not release a version of the product and plan to implement security into a later revision. Retrofitting security mechanisms is extremely difficult, and political and financial pressures usually prevent it from actually happening.

ACKNOWLEDGEMENTS

The author would like to thank Brian Hassick for his prior collaboration and discussions on secure hardware techniques.

REFERENCES

1. D.G. Abraham, G.M. Dolan, G.P. Double, and J.V. Stevens, "Transaction Security System," *IBM Systems Journal*, vol. 30, no. 2, 1991, http://www.research.ibm.com/journal/sj/302/ibmsj3002G.pdf.

2. Aleph One, "Smashing the Stack for Fun and Profit," http://www.securityfocus.com/library/14.

3. R.J. Anderson and M. Kuhn, "Low Cost Attacks on Tamper Resistant Devices," *Security Protocols, 5th International Workshop*, 1997, http://www.cl.cam.ac.uk/~mgk25/tamper2.pdf.

4. R.J. Anderson, "Security Engineering - A Guide to Building Dependable Distributed Systems," John Wiley & Sons, 2001.

5. W. Arbaugh, Wireless Research Web Page, http://www.cs.umd.edu/~waa/wireless.html.

6. F. Beck, "Integrated Circuit Analysis - A Guide to Preparation Techniques," John Wiley & Sons, 1998.

7. M. Bond and R. Clayton, "Extracting a 3DES key from an IBM 4758," http://www.cl.cam.ac.uk/~rnc1/descrack.

8. D. Chaum, "Design Concepts for Tamper Responding Systems," *Advances in Cryptology: Proceedings of Crypto '83*.

9. A.J. Clark, "Physical Protection of Cryptographic Devices," *Advances in Cryptology: EUROCRYPT '87*.

10. B. Dipert, "Cunning Circuits Confound Crooks," *EDN Magazine*, October 12, 2000.

11. J. Dyer, M. Lindemann, R. Perez, R. Sailer, S.W. Smith, L. van Doorn, and S. Weingart, "Building the IBM 4758 Secure Coprocessor," *IEEE Computer*, October 2001, http://www.cs.dartmouth.edu/~sws/papers/comp01.pdf.

12. M. Fisher, "Protecting Binary Executables," *Embedded Systems Programming*, February 2000.

13. M.G. Graff and K.R. Van Wyk, "Secure Coding: Principles and Practices," O'Reilly & Associates, 2003.

14. J. Grand (Editor), et al., "Hardware Hacking: Have Fun While Voiding Your Warranty," Syngress Publishing, 2004.

15. P. Gutmann, "Data Remanence in Semiconductor Devices," *Tenth USENIX Security Symposium*, 2001, http://www.usenix.org/publications/library/proceedings/sec01/gutmann.html.

16. P. Gutmann, "Secure Deletion from Magnetic and Solid-State Memory Devices," *Sixth USENIX Security Symposium*, 1996, http://www.usenix.org/publications/library/proceedings/sec96/full_papers/gutmann/index.html.

17. A. Huang, "Hacking the Xbox: An Introduction to Reverse Engineering," No Starch Press, 2003.

18. IC-ID: Integrated Circuit Identification Web Page, http://www.elektronikforum.de/ic-id.

19. R.G. Johnston and A.R.E. Garcia, "Vulnerability Assessment of Security Seals", *Journal of Security Administration*, 1997, http://www.securitymanagement.com/library/lanl_00418796.pdf.

20. Kingpin, "Attacks on and Countermeasures for USB Hardware Token Devices," *Proceedings of the Fifth Nordic Workshop on Secure IT Systems*, 2000, http://www.grandideastudio.com/files/security/tokens/usb_hardware_token.pdf.

21. Kingpin, "DS1991 MultiKey iButton Dictionary Attack Vulnerability," 2001, http://www.grandideastudio.com/files/security/tokens/ds1991_ibutton_advisory.txt.

22. Kingpin and Mudge, "Security Analysis of the Palm Operating System and its Weaknesses Against Malicious Code Threats," *Tenth USENIX Security Symposium*, 2001, http://www.usenix.org/publications/library/proceedings/sec01/kingpin.html.

23. P. Kocher, "Crypto Due Diligence," *RSA Conference 2000*.

24. P. Kocher, "Hacking Cryptosystems," *RSA Conference 2002*, http://www.cryptography.com/resources/whitepapers/HackingCryptosystems.pdf.

25. P. Kocher, "Timing Attacks on Implementations of Diffie-Hellman, RSA, DSS, and Other Systems," *Advances in Cryptology: Proceedings of Crypto '96*, http://www.cryptography.com/resources/whitepapers/TimingAttacks.pdf.

26. P. Kocher, J. Jaffe, and B. Jun, "Differential Power Analysis," *Advances in Cryptology: Proceedings of Crypto '99*, http://www.cryptography.com/resources/whitepapers/DPA.pdf.

27. P. Kocher, J. Jaffe, and B. Jun, "Overview of Differential Power Analysis," http://www.cryptography.com/resources/whitepapers/DPATechInfo.PDF.

28. O. Kömmerling and M. Kuhn, "Design Principles for Tamper-Resistant Smartcard Processors," *USENIX Workshop on Smartcard Technology*, 1999, http://www.cl.cam.ac.uk/~mgk25/sc99-tamper.pdf.

29. M.G. Kuhn and R.J. Anderson, “Soft Tempest: Hidden Data Transmission Using Electromagnetic Emanations,” *Information Hiding*, 1998, http://www.cl.cam.ac.uk/~mgk25/ih98-tempest.pdf.

30. J. Loughry and D.A. Umphress, “Information Leakage from Optical Emanations,” *ACM Transactions on Information and System Security*, 2002, http://www.applied-math.org/optical_tempest.pdf.

31. T.S. Messerges, "Power Analysis Attack Countermeasures and Their Weaknesses," *Communications, Electromagnetics, Propagation, & Signal Processing Workshop*, 2000, http://www.iccip.csl.uiuc.edu/conf/ceps/2000/messerges.pdf.

32. K.D. Mitnick and W.L. Simon, "The Art of Deception: Controlling the Human Element of Security," John Wiley & Sons, 2002.

33. National Institute of Standards and Technology, "Engineering Principles for Information Technology Security (A Baseline for Achieving Security)," *NIST Special Publication 800-27*, http://csrc.nist.gov/publications/nistpubs/800-27/sp800-27.pdf.

34. National Institute of Standards and Technology, "Security Requirements for Cryptographic Modules," *FIPS Publication 140-2*, http://csrc.nist.gov/cryptval.

35. C.P. Pfleeger, "Security in Computing, Second Edition," Prentice Hall PTR, 2000.

36. PIC Microcontroller Discussion List, "Re: Code protect," Posted April 26, 1995, http://www.brouhaha.com/~eric/pic/84security.html.

37. J.R. Rao and P. Rohatgi, "EMPowering Side-Channel Attacks," IBM Research Center, http://www.research.ibm.com/intsec/emf-paper.ps.

38. B. Schneier, "Applied Cryptography: Protocols, Algorithms, and Source Code in C, Second Edition," John Wiley & Sons, 1995.

39. S. Skorobogatov, "Low Temperature Data Remanence in Static RAM," *University of Cambridge Computer Laboratory Technical Report UCAM-CL-TR-536*, June 2002, http://www.cl.cam.ac.uk/TechReports/UCAM-CL-TR-536.pdf.

40. S.W. Smith and S.H. Weingart, "Building a High-Performance, Programmable Secure Coprocessor," *Computer Networks* (Special Issue on Computer Network Security), 1999, http://www.research.ibm.com/secure_systems_department/projects/scop/papers/arch.pdf.

41. W. van Eck, "Electronic Radiation from Video Display Units: An Eavesdropping Risk?" *Computers and Security*, 1985, http://www.jya.com/emr.pdf.

42. J. Viega and M. Messier, "Secure Programming Cookbook for C and C++," O'Reilly & Associates, 2003.

43. S.H. Weinhart, S.R. White, and W.C. Arnold, "An Evaluation System for the Physical Security of Computing Systems," *Sixth Annual Computer Security Applications Conference*, 1993.

44. S.H. Weingart, "Physical Security Devices for Computer Subsystems: A Survey of Attacks and Defenses," *Workshop on Cryptographic Hardware and Embedded Systems*, 2000.

45. O. Whitehouse, "War Nibbling: Bluetooth Insecurity," October 2003, http://www.atstake.com/research/reports/acrobat/atstake_war_nibbling.pdf.

46. xda-developers.com, JTAG for the XDA Web Page, http://xda-developers.com/JTAG.

47. B.S. Yee, "Using Secure Coprocessors," *Carnegie Mellon University*, 1994, http://citeseer.nj.nec.com/yee94using.html.

48. J. Young, TEMPEST Documents Web Page, http://cryptome.org/nsa-tempest.htm.

FURTHER READING

The secure hardware design and computer security fields have a long and interesting history, and a great deal of work and research has been performed in these areas. Besides the references that appear at the end of this paper, here are additional sources of insight:

System and Product Development

- K.R. Fowler, "Electronic Instrument Design - Architecting for the Life Cycle," Oxford University Press, 1996.

- C. Salter, O. Saydjari, B. Schneier, and J. Wallner, "Toward a Secure System Engineering Methodology," *New Security Paradigms Workshop*, September 1998, http://www.schneier.com/paper-secure-methodology.pdf.

Attacks and Prevention

- R.J. Anderson and M. Kuhn, "Tamper Resistance - a Cautionary Note," *The Second USENIX Workshop on Electronic Commerce*, 1996, `http://www.cl.cam.ac.uk/~mgk25/tamper.pdf`.

- C. Clavier, J.S. Coron, and N. Dabbous, “Differential Power Analysis in the Presence of Hardware Countermeasures,” *Workshop on Cryptographic Hardware and Embedded Systems*, 2000.

- W.W. Fung and J.W. Gray, "Protection Against EEPROM Modification Attacks," *Information Security and Privacy*, 1998.

Cryptographic Coprocessors

- W. Arbaugh, D. Farber, and J. Smith, "A Secure and Reliable Bootstrap Architecture," *IEEE Security and Privacy Conference*, 1997, `http://www.cs.umd.edu/~waa/pubs/oakland97.pdf`.

- P. Gutmann, "An Open-Source Cryptographic Coprocessor," *9th USENIX Security Symposium*, 2000, `http://www.usenix.org/publications/library/proceedings/sec2000/gutmann.html`.

Reverse Engineering

- A. Huang, "Keeping Secrets in Hardware: the Microsoft XBox Case Study," *Massachusetts Institute of Technology AI Memo 2002-008*, May 2002, `http://web.mit.edu/bunnie/www/proj/anatak/AIM-2002-008.pdf`.

- R. Russell (Editor), et al., "Hack Proofing Your Network, Second Edition," Syngress Publishing, 2002.

Digital Rights Management and Trusted Computing

Although not directly discussed in this paper, Digital Rights Management (DRM) schemes, typically a combination of software-based controls and hardware-based encryption key management, are implemented to protect digital content for widespread distribution (through media such as DVDs or online providers such as Apple iTunes Music Store). DRM and content protection are relatively new research areas and have become the focus of many technical and political discussions.

- Electronic Frontier Foundation, Intellectual Property: Digital Rights Management Systems & Copy-Protection Schemes Archive, `http://www.eff.org/IP/DRM`.

- Microsoft, Windows Media: Digital Rights Management, `http://www.microsoft.com/windows/windowsmedia/drm.aspx`.

The Trusted Computing Group (TCG) is an alliance of major computer hardware and software manufacturers, including Microsoft, Intel, IBM, HP, and AMD, which was formed to "develop and promote open industry standard specifications for trusted computing hardware building blocks and software interfaces across multiple platforms." The TCG standard specifies a key hardware component called a Trusted Platform Module (TPM). TCG was primarily conceived for DRM purposes and to essentially create a computing platform on which the application software cannot be tampered with and such applications can communicate securely with their creators and with each other.

- Trusted Computing Group, `http://www.trustedcomputinggroup.org`.

- R. Anderson, "Trusted Computing" FAQ, `http://www.cl.cam.ac.uk/users/rja14/tcpa-faq.html`.