

# Designing for Security – Why Software Isn't Enough

**Author:**

Greg Powell, President and CEO,  
Valicore Technologies

**Synopsis:**

*This article exposes some of the issues that are often overlooked when designing today's security architectures and provides a discussion of high-integrity security solutions that create a hardware-enforced security environment.*

In today's digital world, remote access is fast becoming the norm. Voice, video and high-value data routinely travel over wired and wireless connections. Consumers download videos and music, employees access corporate networks, cell phones deliver the Internet, families watch pay-per-view programming, vacationers gamble poolside at Las Vegas casinos, and the list goes on. Virtually every piece of information can be accessed from somewhere else. And that's where the problem lies. For those so inclined, improperly protected data is there for the taking. It's simply a matter of motivation.

Consider, too, that as new applications come online—wireless remote gambling, electronic wallets, cell phone applications, and the increasing use of Digital Rights Management (DRM) connected to audio, video and online streaming services for example—a whole new set of security concerns are being introduced.

The purpose of this article is to expose some of the issues that are often overlooked when designing today's security architectures and provide a discussion of high-integrity security solutions that create a hardware-enforced security environment. As engineers, we are responsible for system security. Therefore, we must understand the techniques for implementing and integrating solutions that match the unique requirements of the system as dictated by the value of the data to be protected. Fortunately there are excellent building blocks available for designing high-integrity, embedded environments.

**The Problem**

According to the 2005 FBI Computer Crime Survey, nearly nine out of ten organizations experienced security incidents in 2005. Over 64 percent of respondents incurred a financial loss as a result of computer crime. Consider:

- Over the course of four years, a Romanian hacker (working with a group of Americans) bypassed the password security system of Ingram Micro, a major electronic products distributor. Once the system was compromised, they ordered computer equipment for shipment to various fictitious people and businesses. Finally indicted in 2004, the group defrauded Ingram Micro out of \$10 million worth of equipment.
- In March of 2005, Norway's most notable hacker, DVD Jon, wrote a Python-based program allowing the download of purchased files from the iTunes Music Store without encryption. The same day Apple released a patch, DVD Jon distributed a new program that circumvented it.
- Philip Cummings used his position at his company's help desk to gain unauthorized access to the codes that other companies used to check consumer credit. He then sold credit reports that included social security numbers, credit card numbers and other vital personal information to criminals who used it to defraud victims across the country. More than 30,000 people were affected and losses totaled more than \$2.7 million.
- Sony developed anti-piracy protection which was added to all its CDs. After eight months of use, it was found to create a security breach in the computers it had been used on. The program actually installed a "rootkit" programming tool used by hackers to hide viruses on hard drives. Over 4.7 million CDs had to be recalled.

While there are plenty more examples, these serve to highlight the range of security breaches and the scope of potential loss when systems are compromised.

## Understanding the Threat

Companies must understand the specific threats to their system in order to develop a solution that provides the right level of security. Keep in mind that it's not just bank and government databases that are at risk, but non-commercial entities that store sensitive information that can be sold or incorporated into competitive, revenue-producing products. In the case of remote access, compromised authentication can enable illegal access to private networks and private data. Compromise of pay television services requires real-time access to delivered content. And when the security of a DRM system is penetrated, control over the audio or video information can be permanently lost.

The value of the protected data is key to understanding a hacker's motivation and, therefore, applying a suitable solution. While no system is perfectly secure, the objective is to create a security environment that requires more time and money to compromise than the payoff would yield. Compromise can lead to identity theft, corporate espionage, exposure of secret government documents, access to intellectual property, the ability to create and sell a cloned product like a set-top box, and other rewards. Each has a level of monetary value. And the higher that value, the greater a hacker's motivation. Where there is little reward in hacking a system, a simple PIN may be adequate protection. But as the value rises, so does the attraction to hackers who have the time, talent, tools and techniques to penetrate the system. In some cases, such as cable cards and set-top boxes, the value to the hacking community can be in the hundreds of millions of dollars. Therefore, the robustness of the security solution must be designed accordingly.

## Building a Chain of Trust

Building a chain of trust between the remote devices that access the content is critical. In fact, it's often more important to authenticate the devices that are accessing the network than the people using them. This is particularly true in applications such as cable and satellite TV where the companies need to trust set-top boxes encrypting and displaying program content. Similarly, a company like Apple wants to know that when data is being downloaded to a device, it is being put on a valid iPod. In the case of a corporate network, the ability to trust the devices man-

aging the content or accessing the network is vital. This can be complicated by the fact that multiple vendors may have devices accessing a central repository. The weakest device can compromise the system for everyone.

## Software-only Solutions are Inadequate

Software-only implementations are inadequate—particularly where the incentive to hack is high. These solutions often take many man hours and considerable expense to develop and implement. But because the code, encryption algorithms and keys are exposed, they can be relatively easy to clone and hack—especially with the many reverse engineering tools that are readily available for downloading on the Internet. DRM hacks such as those by DVD Jon are prime examples. After lengthy and expensive development and implementation, the software solution was hacked in a matter of weeks. And once audio and video are available in clear form, it's virtually impossible to recover.

Even for embedded devices it becomes essential to isolate the security-critical processing from the rest of the application. Through the physical interfaces where commands and data are delivered to and from the embedded device, hackers will attempt to exploit system and implementation flaws. Once a vulnerability is found, security components that are not isolated become susceptible to attack. As all device manufacturers also know, however, the cost of producing the product cannot be ignored. It may be reasonable from a security-centric view that multiple processors with completely separate memory and storage would sufficiently isolate secure processing from the general application code and processor. A well-defined interface between the two can offer a great deal of protection from software hacks that may compromise the application responsible for communicating with the outside world. Unfortunately price competition in the market usually makes it impossible to implement such complete redundancy. In many cases, board real estate inside the embedded device is scarce and would not allow for the extra ASICs and memory required for such additional duplicate components.

As companies look for system-on-chip solutions that will solve their need for isolating security, it becomes important to

understand the security techniques that will protect different security systems. The challenge is then to determine what technologies are available in processors today. Companies developing their own system-on-chip processors and companies using available processor solutions will benefit by understanding some key security features that can be incorporated in hardware to greatly strengthen the security of their system.

## Security Techniques

To provide privacy, authentication and non-repudiation, security must include a combination of hardware and software. Even if a software hacker can reverse engineer the portion of software to which he has access, he still doesn't have insight into what's going on in the hardware.

For purposes of this article, the focus is on single system-on-chip solutions or single processor on a printed circuit board (PCB) solutions. The reference to the security domain is the portion of the ASIC or PCB that contains the secure hardware and software components. Further, we incorporate the use of ARM processor. Widely used and licensed by major semiconductor manufacturers such as Intel, Freescale (Motorola) and Philips, ARM offers a wide range of processor based on a common architecture. ARM 32-bit embedded RISC microprocessors combine high performance with low power consumption and low system cost.

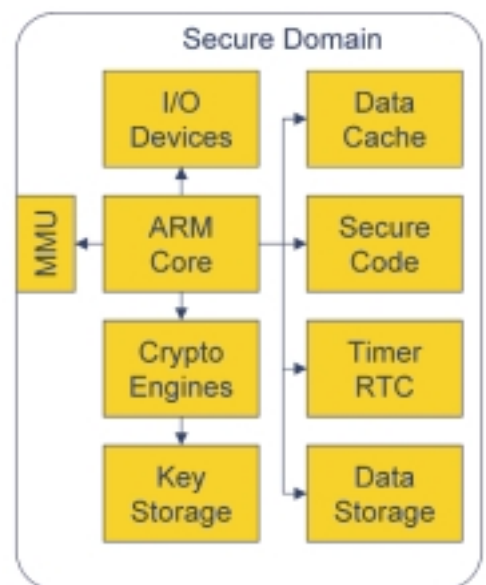


Figure 1: Secure Domain Diagram

The ARM processor range provides solutions for:

- Open platforms running complex operating systems for wireless, consumer and imaging applications
- Embedded real-time systems for mass storage, automotive, industrial and networking applications
- Secure applications including smart cards and SIMMs

**Processor and Feature Implementation**

Understanding the processor and how it handles security techniques is integral to determining what features can be implemented. Most notable are secure interrupt handling, the ability to run secure code, access to a store of secure data and keys, and controlled access to secure devices that are tied directly to the processor.

- **Secure Interrupts** – A secure interrupt processing feature prevents interrupts from non-secure devices suspending security processes. Hackers often attempt to force non-secure interrupts to gain access to secure content on the bus or in system caches. By separating the interrupts, processor cores can prevent direct access to secure data or the disruption of critical security processing.

- **Secure Cache** – By utilizing two separate cache areas in the processor—secure and non-secure—non-secure code will never have access to memory areas where secure code is running.

This eliminates the possibility of non-secure code having access to secure data that may otherwise be left behind in a data cache that was not cleared.

- **Controlled Bus** – Most designs have other FPGAs or secure devices on the local bus that are used to implement security features such as memory for cryptographic algorithms or key storage. Like the cache, sharing the bus would give non-secure code and devices access to secure devices. As a result, processor are now starting to establish separate busses for secure and non-secure data or designing in the ability to lock out the bus while the processor is in a secure mode of operation.

**Secure Code**

Since many hacks start by extracting the firmware and examining it for exploitable flaws, encryption is necessary to protect code privacy. Encryption allows the code to be hidden and prevent reverse engineering. Possibly more important is the ability to sign the code so that authentication is required prior to execution. The objective is to ensure that no one has the ability to change the code as well as to ensure that it is the original code. This feature gives the security domain the ability to verify the system before start up. It can also perform self tests, as required by security standards such as FIPS 140-2, prior to start up, including the verification and initialization of programmable hardware devices.

**Cryptographic Engines**

Cryptographic engines can be implemented in software, hardware or a combination of both. However, when algorithms are implemented in software, the keys are also available in software. If a hacker is able to reverse engineer the algorithms, chances are he'll also have access to the keys.

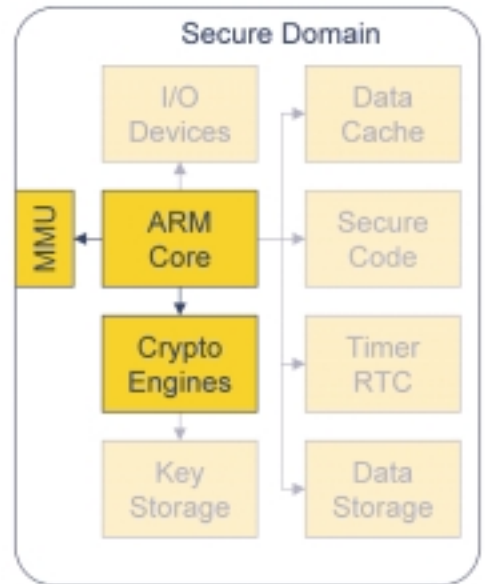


Figure 4: Crypto Engines Diagram

Protecting the keys may be one of the most important factors in system design. To trust a device, you must trust that the keys have not been compromised. Hardware encryption is the best solution as it provides far better performance and is much more difficult to hack. More importantly, hardware encryption provides the ability to protect keying information. By storing encryption keys where they can only be accessed by the hardware, software running on the general purpose processor cannot access the keys. A hacker would have to reverse engineer the processor cores at run time to gain access to the keying material. This is the primary reason why, from a security standpoint, that cryptography should be implemented in hardware (Figure 5 on next page).

Another useful design consideration is to provide only the algorithms needed for encryption or decryption. For example, if the system only needs to decrypt prior to execution, don't include the encryption algorithm. This would prevent a hacker from modifying the code in an attempt to use the engine to encrypt data or code and pass it off as original, authentic content.

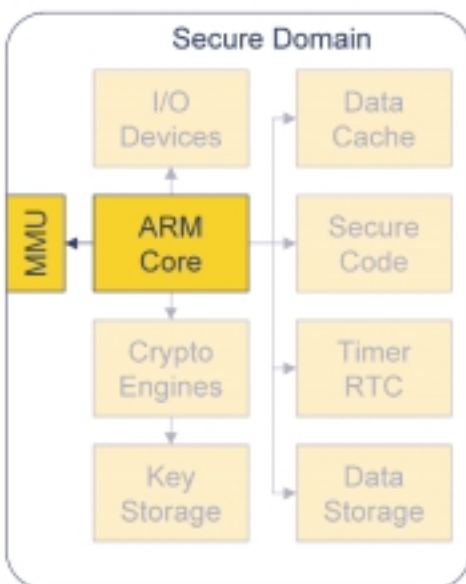


Figure 2: ARM Diagram

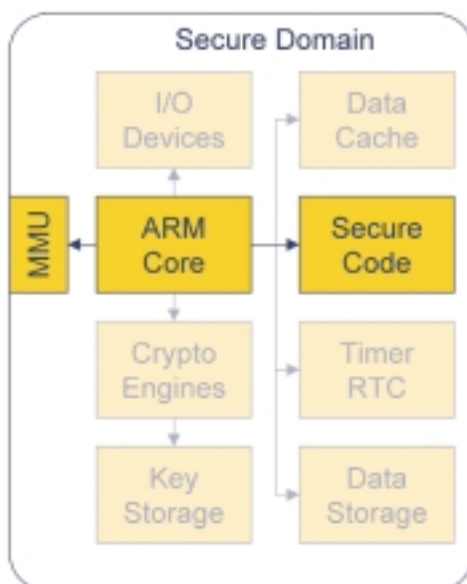


Figure 3: Secure Code Diagram

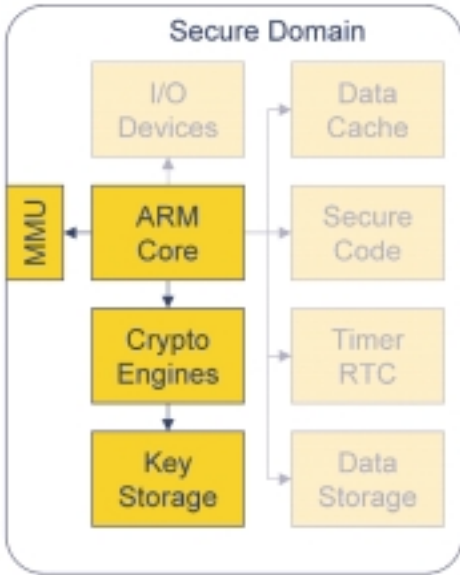


Figure 5: Key Storage Diagram

It's also essential to stick with well-known algorithms that have been developed by experienced cryptographers and tested over many years of use. Untested proprietary algorithms can create undue exposure to security breaches.

**Secure Data Caching and Storage**

Encryption algorithms and keys aren't the only information that must be protected. System control information, user IDs, error logs and personal information should also be encrypted. Modification of privileges can give a user access to sensitive information. And access to event logs can enable hackers to find a system's weaknesses. In many industries such as banking and health care, encryption of this sensitive

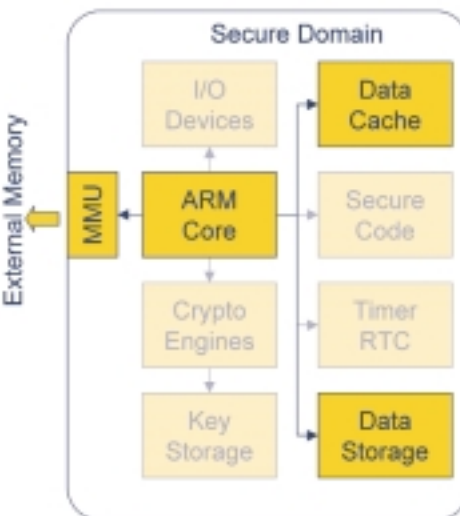


Figure 6: Data Cache/Data Storage Diagram

information is required by law. For that reason, a thorough understanding of what information is considered critical for your industry is a must.

**Trusted Busses**

Input and output to secure devices, such as fingerprint scanners, PIN pads, alarm signals and warning lights, is often a necessity. A trusted I/O path can build trust in the transferred data. The more isolated those busses are for communications to the I/O devices, the more it separates hackers, viruses or rogue software from gaining control of the bus in an attempt to conceal warnings and alarms or to capture user input.

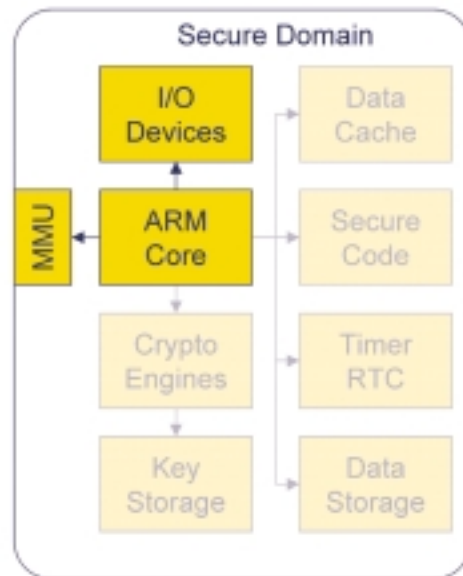


Figure 7: I/O Devices Diagram

**Choosing the Right Technique**

It should be clear that the correct choice of security techniques depends on the type of application and the expected security threats. The table below shows a sample of the types of techniques and how they differ

	DRM	Access	Cell Phone	eWallet	Pay TV	Gov. Radio
Secure Code	X		X		X	X
Key Storage	X	X	X	X	X	X
Secure Data	X		X	X		X
I/O Devices		X				X
HW Crypto	X				X	X

Figure 8: Security Techniques Diagram

based on application. While not a comprehensive list, these techniques have been successful in thwarting real attacks and preventing unauthorized access to sensitive data. Different systems require different techniques. Evaluate and incorporate techniques that are advantageous to your system and its security environment. Conversely, don't spend time or money on those that have little benefit.

**Technologies**

What technologies are available today to implement these techniques?

There are a variety of technologies that are becoming available for use in custom ASIC designs and custom board designs. Some key enabling technologies specifically designed for securing consumer products (such as mobile phones and PDAs running operating systems, such as Symbian OS, Linux and Windows CE) can be found in ARM TrustZone®. As with other available security technologies, TrustZone focuses on features that allow for reliable implementation of critical security applications and services such as secure code execution and the protection of critical secrets such as keys. We have highlighted five technologies, some of which can be found in the TrustZone architecture, that are important building blocks for implementing the security techniques outlined in this article (Figure 9 on next page).

Secure Interrupt Controller (SIC) – The SIC separates secure interrupts from non-secure interrupts. It also stops the chosen source from propagating to the non-secure interrupt controller. Only secure interrupts are permitted to transition to secure mode and pass control to the defined secure interrupt handler. This is essential to make sure non-secure code does not get control of the processor and gain access to secure



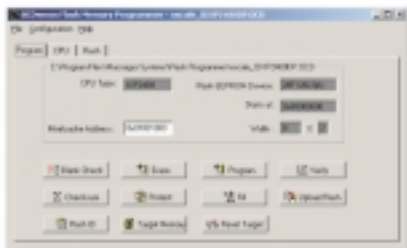
# OCDemon™ MACRAIGOR SYSTEMS

JTAG interfaces for all your needs ... connect via USB, Ethernet, serial or parallel ports ... test, debug, and flash program all with one device.

Prices start at \$150.



Free software tools including debuggers, C and C++ compilers, assemblers, linkers, and full GUIs. Just check out our web site.



Need to program FLASH devices in circuit? How about production line test? We have the software you are looking for starting as low as \$100.

## MACRAIGOR SYSTEMS

PO Box 471008  
Brookline Village, MA 02447  
Sales: (206) 855-9269

[www.macraigor.com](http://www.macraigor.com)

	SIC	OTP Memory	Secure Cache	MDPC	Secure Debug
Secure Code	X		X		X
Key Storage		X	X		X
Secure Data	X	X	X		X
I/O Devices	X			X	X
HW Crypto	X			X	X

Figure 9: Technologies Diagram

peripherals or memory. Secure interrupts act as non-maskable interrupts when the core is in non-secure state.

One Time Programmable (OTP) – OTP on-chip memory can be used for writing unique IDs and protecting keying material, like root public keys that will not change. Once written, they can not be modified or replaced.

Secure Cache – Secure Cache is an important feature for separating a secure execution environment from a non-secure environment when only one processor is available to handle both sets of code. The ARM processors and TrustZone are built around their ability to execute signed and secure code that is separate from the non-secure code. Controlled by the MMU, the internal cache is managed between secure and non-secure modes ensuring that non-secure code can not access data left in the cache after leaving the secure runtime environment. In secure state, access security is determined by an additional security bit in the processors' MMU page tables. This bit is crucial to ensure that cache coherency is maintained between secure and non-secure states.

Memory and Device Protection Controller (MDPC) – The MDPC provides a protected control interface for tying other peripherals such as memories, fingerprint scanners or external cryptographic engines into the ARM processor via a trusted bus. Providing a high degree of system flexibility and configurability, it enforces which devices are accessible by the secure domain and secure code. The MDPC registers must be memory-mapped into a

secure area of memory and are generally only configured at boot-up or even at the time of manufacture through the use of configuration values set permanently in OTP.

Debug Security – One of the problems that many security devices have is disabling the JTAG debugging interface after development. ARM's banked debug registers can be disabled in stages via hardware PINs after the secure environment is ready for production. The system allows for debugging of non-secure code, but locks out access to internal memories and secure code. Evaluating a processor's JTAG configurability is critical to architecting an ASIC or processor board that can permanently disable debug interfaces on production products.

### Secure Embedded Solutions are Available

ARM processor and the TrustZone suite are not the only available solution. They are, however, good examples of invaluable building blocks for designing and implementing high-integrity, embedded security systems. They provide the needed features and techniques to match the level of security to your application. Remember, by raising the level of resources required to hack the system to a height greater than the potential payoff, you can effectively discourage hackers based on economics. With only this one objective in mind, you can achieve real security, produce better products and experience fewer problems once your products are released into the marketplace.



## Back a winner!



Atmel's ARM<sup>®</sup>-based 32-bit microcontrollers are winners. They have already picked up three awards from readers of industry leading magazines\*. Why? Because they give you exactly what you want. On-chip Flash memory, USB connectivity, DMA to eliminate internal bottlenecks, advanced interrupt handling. All this at the lowest possible power consumption and unit price. Plus code compatibility across the entire product family. So, make your application a winner by backing a winner: Atmel's AT91SAM Smart ARM-based microcontrollers.

\* EEProductCenter Ultimate Product (Processor & Memory) for Q4 2004 and again for Q4 2005, Embedded Control Europe Gold Award (Micros & DSP) for H2 2005



Check out Atmel's AT91 solutions at [www.atmel.com/products/at91](http://www.atmel.com/products/at91)



Everywhere You Are<sup>®</sup>

© Atmel Corporation 2006. All rights reserved. Atmel<sup>®</sup>, logo, combinations thereof, Everywhere You Are<sup>®</sup> and others are registered trademarks of Atmel Corporation or its subsidiaries. ARM<sup>®</sup> is the registered trademark of ARM Ltd. Other terms and product names may be trademarks of others.