Secure Processors for Embedded Applications

Black Hat DC 2007

February 28, 2007

James D. Broesch

jbroesch@UCSD.edu

+1-858-964-6842

Introduction

- Most modern security work is based on either protecting or cracking either the communications network or the vulnerabilities of the (generally commercial) OS on the target system.
- This is not surprising since the majority of modern systems make use of conventional processors running standard operating systems and communicating over standard channels such as Ethernet or WiFi.
- While much good work has been done in these areas, to some extent any such effort must be seen as ultimately a dead-end for obtaining a truly secure system.

Introduction (continued)

- The first and most obvious problem is that such efforts do not physically protect the system.
- Numerous examples have made the news of stolen laptops, stolen computers, etc. containing sensitive – and even classified – information.
- To some extent, these problems can be mitigated by good policies and procedures: using encrypted disc partitions, VPNs, etc.
- However, these all assume that proper procedures are followed by the users and that no latent vulnerabilities are present (for example, easily broken passwords, passwords stored in the clear, etc.)

Thus the motivation to use secure processors is growing for a variety of reasons

- Protection of IP
  - Algorithms
  - Cores
  - Configuration Files (FPGAs)
- Protection of design information (prevent reverse engineering)
- ITAR (International Traffic in Arms Regulations) requirements may require secure processors to meet export restrictions
- Federal Information Processing Standards (FIPS / Common Criteria) will generally be mandated for many kinds of commercial transactions
- Protection of key data where simple storage encryption is not adequate
- Prevent exploit vulnerabilities (rigged gamming machines, vulnerable ATMs, etc.)

The (painful and partial) Solution

- In order to build a truly secure system it is necessary to start from the ground up.
- This is why this talk addresses embedded systems; the basic principles are applicable to both standard computers systems and embedded systems. However only a system designed from the first principles for security is likely to truly achieve a high level of security.
- Like justice, there is no such thing as absolute security. However, in this talk I will be discussing techniques that both show you how to get very close to an absolutely secure system and how to judge how vulnerable your system may be.

First Principles

- Start with secure silicon. This means chips from a trusted fab. No “back-doors”, no “un-documented” features, no “reserved for factory test.”
- Write your own kernels and applications in a secure (two person check) environment.
- Or better yet do not use conventional programming techniques. Directly implement your functions in hardware. This still means using a secure (two person check) design environment.
- Limit access to all design information.
- Secure (encrypt) all software and secure (physically protect) all hardware.
- Have a strong and reliable key management system in place.

Threats

- Reverse Engineering; unauthorized copying of the design
- Exposing the operation of critical technologies
- Developing counter measures to the mission of the system
- Spoofing of the system (Information Assurance)

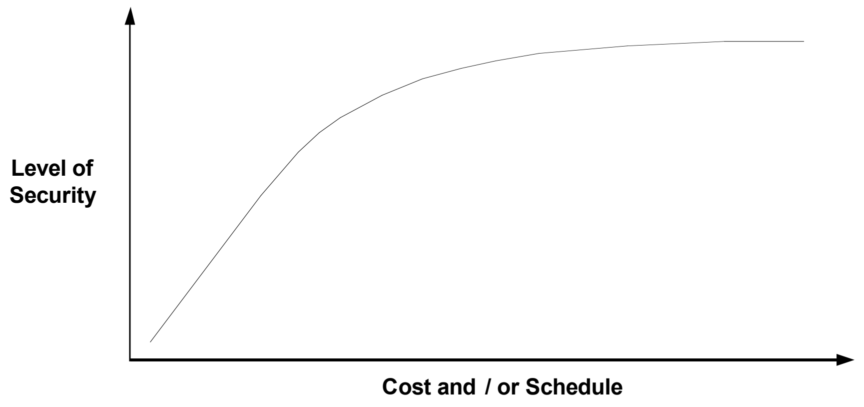
Threat Vectors

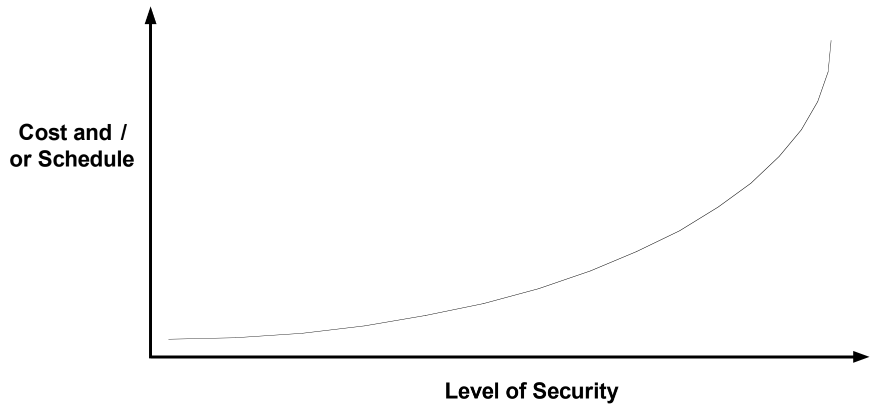
- Inspection
  - ➢ Module penetration
    - X-Rays
    - Thermal
    - Sonic
  - ➢ Circuit Layout
  - ➢ Power Analysis
  - ➢ Deliding Devices
- Power Analysis
- Thermal Analysis

Practical Considerations

- Few programs can afford the time or money to build a secure system from first principles.
- There are usually practical constraints:
  - Mobile devices make physical security more difficult if not impossible.
  - The design complexity of modern systems often makes it impractical to use fully custom hardware and / or software.
  - Most systems must conform to commercial standards such as TCP/IP, etc.

Security Evaluations





The usual exponential curves apply to security as to most other engineering activities:

- There is a point of diminishing returns.
- There is an exponential cost increase for higher levels of security.

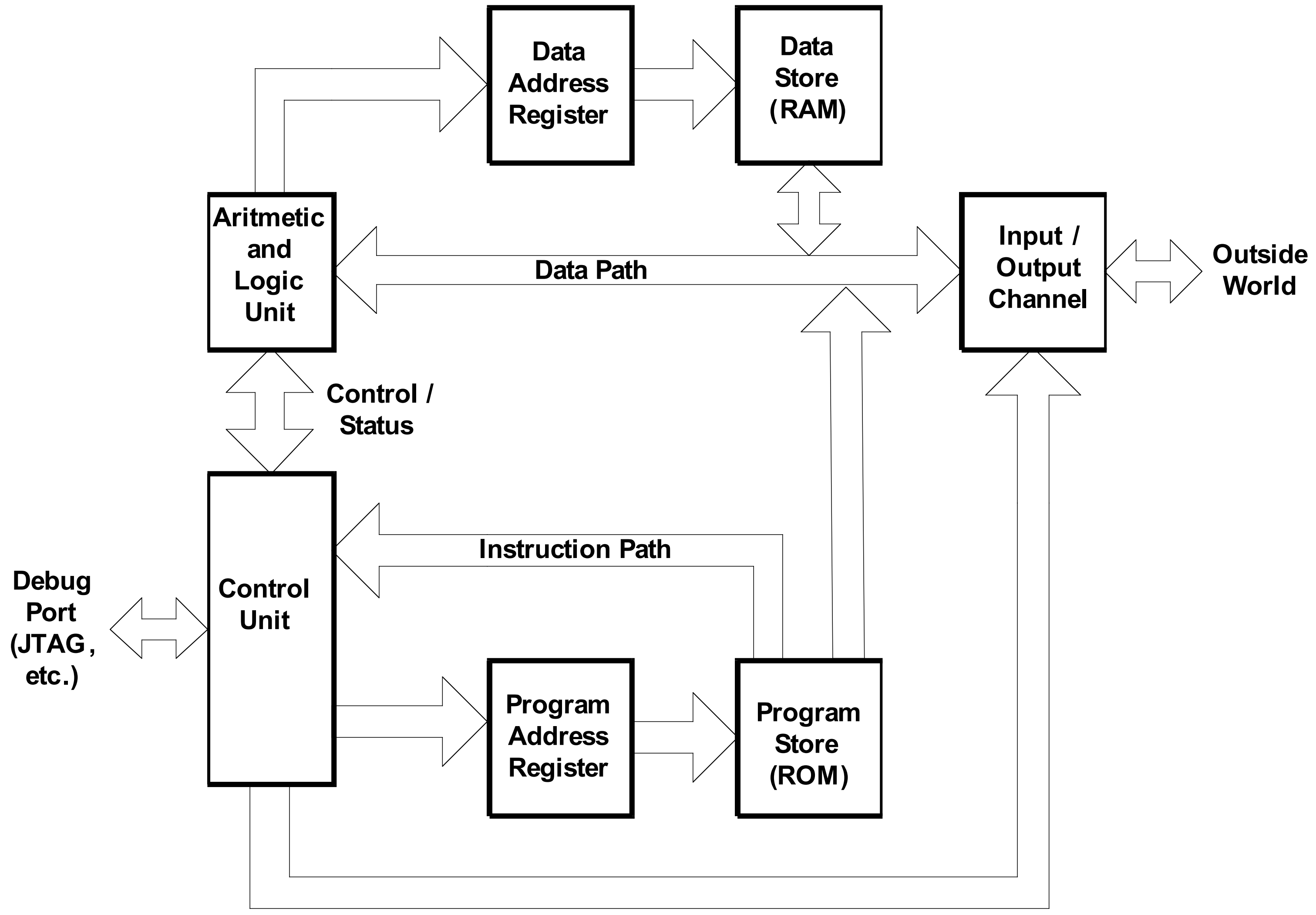
Security Evaluations (continued)

- So, the optimal system design becomes a tricky tradeoff between technical feasibility, cost, schedule, level of effort, and required degree of *practical* security.
- Key questions / motivations:
  - Simple paranoia
  - IP / design protection
  - Protection of sensitive (i.e. trade secrets, business plans, etc.) data.
  - Protection legally liable data (i.e. medical records (HIPA), financial transactions (ATMs, SOX), etc.)
  - Classified information (which, needless to say, we will *not* be discussing.)

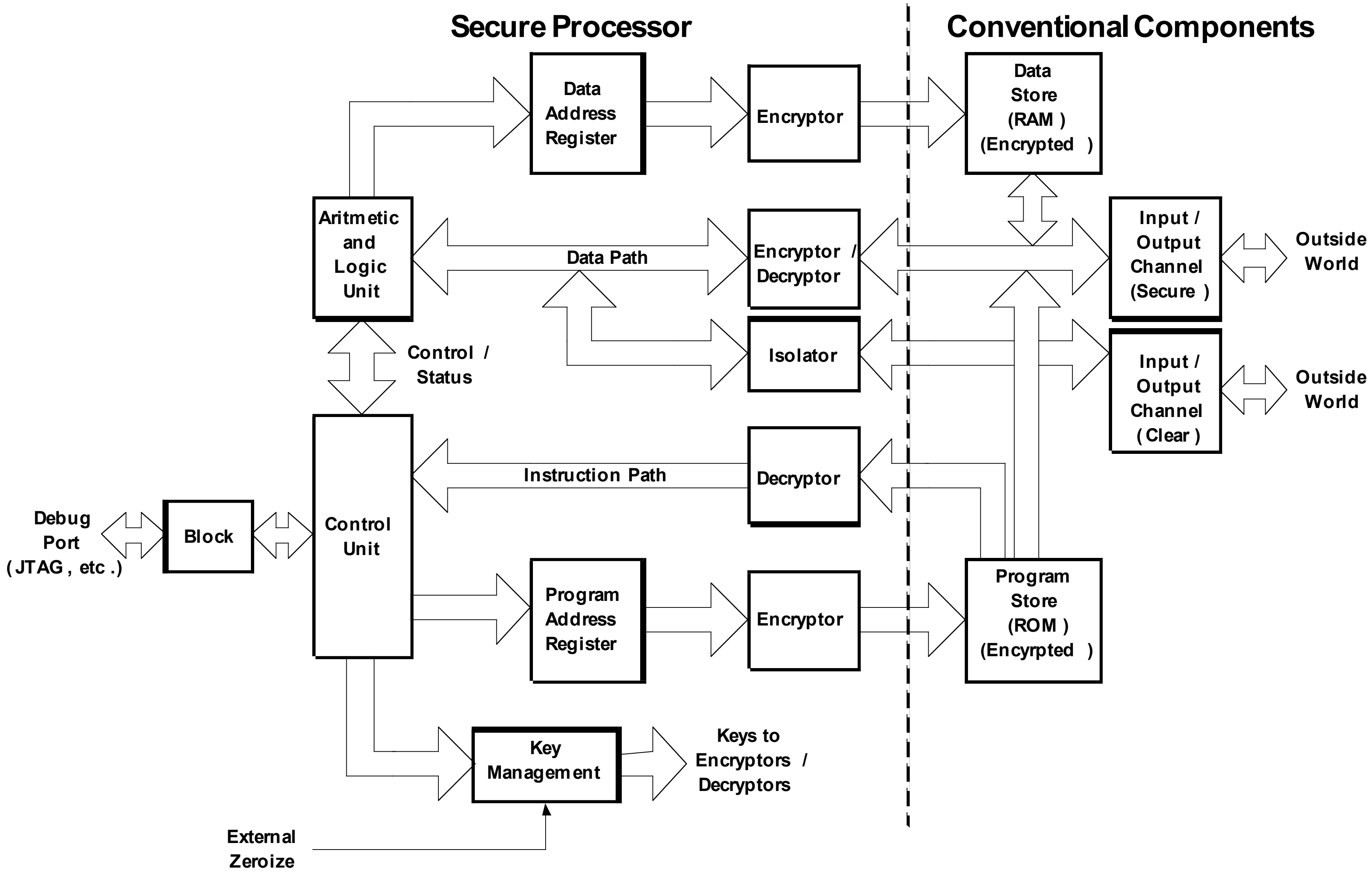
So what exactly is a Secure Processor?

- Like may things in life, a secure processor means different things to different people.
- The key characteristics will generally include, but are not necessarily limited to:
  - Some type of anti-tamper (AT) / intrusion detection
    - This may include forensic enhancements
    - And it may include information assurance (IA) enhancements.
  - The ability to execute secured (encrypted) programs
  - The ability to protect data from unauthorized access
  - The ability to protect code from unauthorized access
  - A secure communications channel (secured I/O)
  - Some type of isolation from non-secure I/O channels
  - Security may also mean high-reliability: is a processor secure if it is susceptible to routine failures?
  - It may mean high performance. Encryption, authentication, etc. can take up a significant amount of processing power.
  - Accelerators for encryptions may be available.

First, lets look at a conventional Harvard Processor



Now, let's see what a secure version might look like:



Architectural features of the Secure Processor

- First, there is a new section for key management
  - Keys may be externally loadable or “hardwired” in.
  - Externally loadable keys can generally be “zeorized” (erased). Often this can be done internally or externally.
  - Hardwired keys are generally not visible under any (reasonable) conditions to the outside world. Thus they are in some sense “more secure.”
- Intrusion detection / prevention is present. Keys may be zeroized, memory erased, or other responses to an attempt to penetrate the system.

Architectural features of the Secure Processor (continued)

- Note the “double encryption”
  - The data is encrypted before it is stored in external memory.
  - Similarly, the address in also encrypted.
  - Assuming each processor is keyed individually, breaking the encryption on the data alone will not compromise other systems: the data will still be “scrambled”.
- For our example system, the program is also double encrypted. In this case, it is assumed that an external PROM’s instructions and constants are encrypted. This information is then stored in the PROM using encrypted addresses.

Architectural features of the Secure Processor (continued)

- Note that there is both a *secure* I/O channel and a *non-secure* (clear) I/O channel.
  - ➢ The strength of the security of the Secure Processor is directly dependent upon how well these two channels are isolated.
  - ➢ The easiest place to attack a secure processor is generally at this point of isolation!
  - ➢ If this isolation can be penetrated all data transaction can be monitored “in the clear”

There will be some protection against attack by power analysis (SPA, DPA), Differential Electromagnetic Analysis (DEMA), etc.

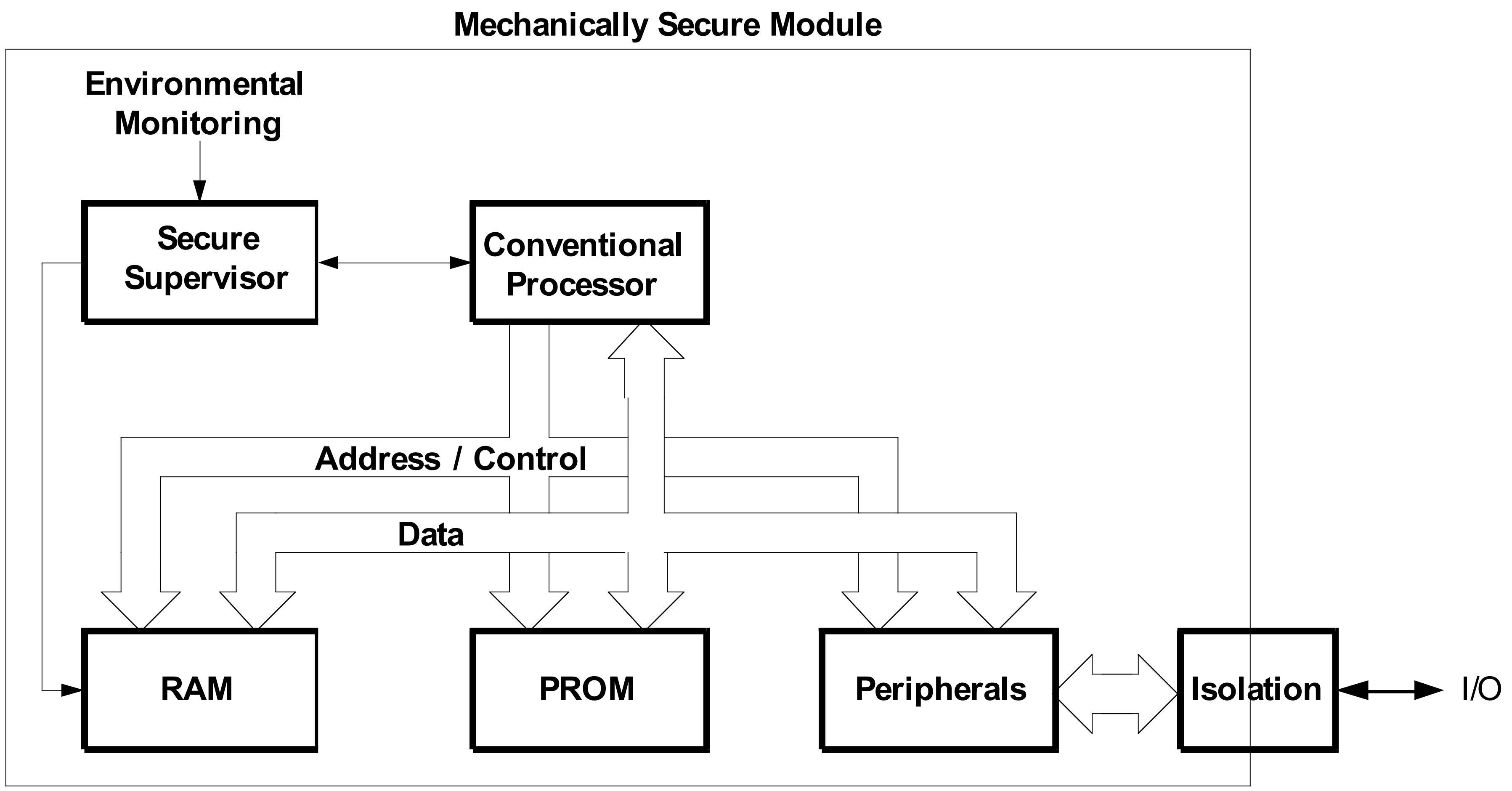
Classes of Secure Processors

- Depending upon the requirement, there are a variety of Secure processor options.
- The strongest, though most costly and difficult, is a full custom ASIC. Though it should be remembered that unless the process described earlier are followed, a poorly executed custom ASIC may be less secure than a standard product.
- Security enabled designs using conventional processors.
- Legacy microcontrollers / microprocessor that have been modified for security.
- Programmable architecture devices designed for security.
- Systems on a Chip (SoC)
  - These can be hard systems (fixed configuration)
  - Or they may be Field Programmable Gate Arrays with security features.
- Modern RISC processors or cores designed for use secured systems.

Simple Examples

- The simplest is a conventional microcontroller that simply has some built in security features. For example, the Rabbit 4000 [1]:
  - This is not really a secure processor in any true sense of the word. However, it has hardware and software that optimize its 8-bit microcontroller architecture for the 32 bit operations often required for encryption
- Another example is the Microchips' dsPIC30F
  - Again, not truly a secure processor. However, libraries are available to make the dsPIC30F's hardware usable for encryption and decryption.
- Although not truly secure processors, these devices can be enhanced with devices such as Dallas Semiconductor's DS3600 Secure Supervisor.
  - Provides tamper detection
  - Provides secure key storage

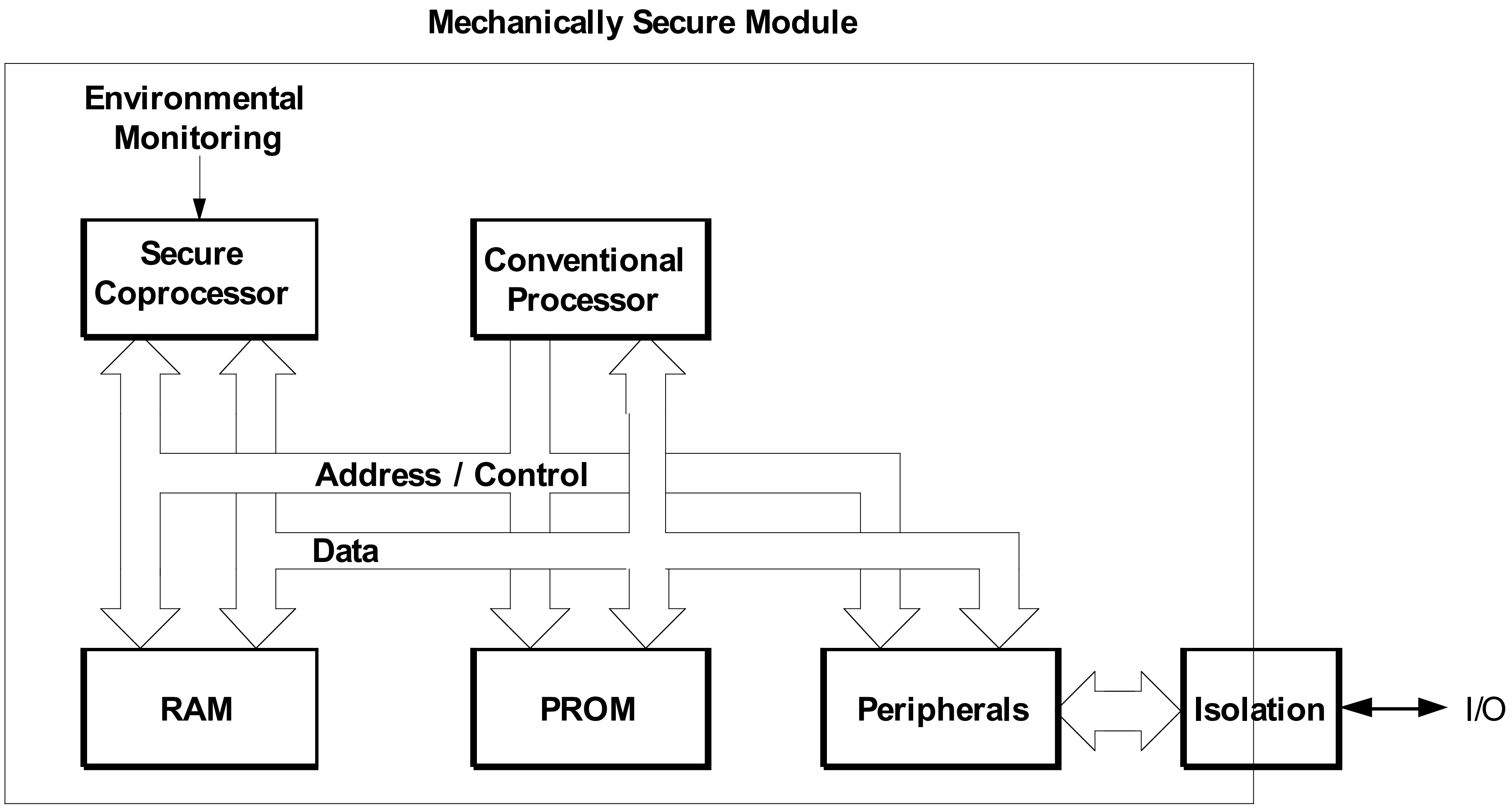
von Neumann System Secured with a Monitor and Physical Security



Legacy Processors that have been secured

- As general rule, conventional processors that have been “secured” are based on legacy processor cores that are then wrapped in a security shell.
- These are true secure processors.
- They are generally targeted at FIPS / Common Criteria applications.
- Atmel’s AT90SC12836RCFT for example is based on the AVR 8/16 bit processor. The processor is rated at 1 MIPS
- Maxim’s DS5250 is based on the 8051 Microcontroller. Performance is 4 clock cycles per instruction with a maximum 25 MHZ clock rate.
- These processors can be combined with high performance non secure processors to form a (reasonably) secure embedded processor system [2]

von Neumann System Secured with Secure Coprocessor and Physical Security



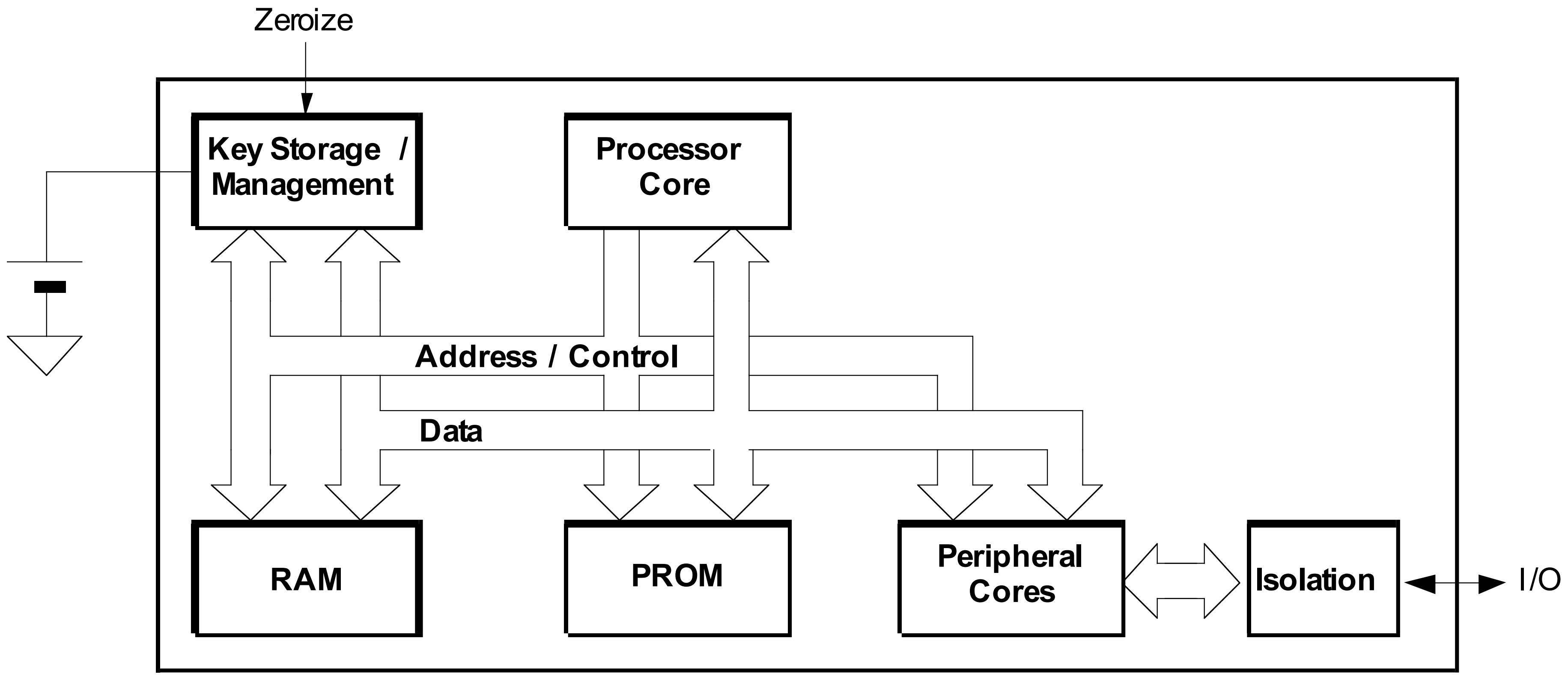
Trades of Securing Conventional Systems

- Advantages:
  - The system can be designed with conventional processors, components, and methodologies.
  - This provides the maximum in flexibility of implementation options.
- Disadvantages:
  - Requires careful mechanical design of the module.
  - N+1 Problem: given enough units, the attacker will eventually find a way into the system.
  - Once the hacker is in, the system is completely exposed.
  - This last statement can be challenged: the secured monitor or secured processor may well still be secure. However, the attacker will have complete visibility of all conventional processor traffic. Therefore, they will probably be able to obtain anything of interest they want (depending upon the application.)

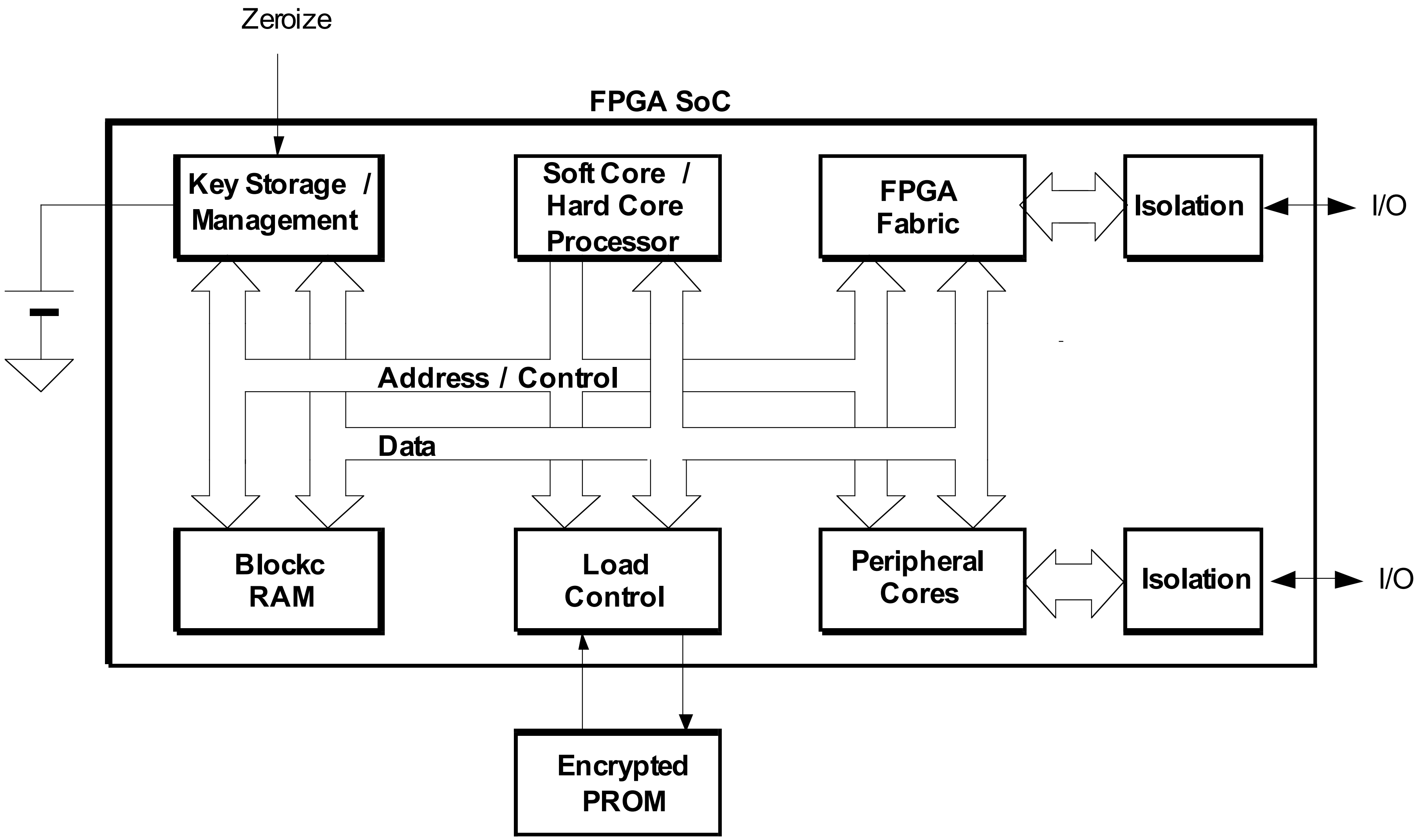
System on a Chip (SoC)

- Modern Systems on a Chip offer significant opportunities for implementing Secured Embedded Systems.
- All components are enclosed on a single die. This alone may be enough meet the system requirements.
- Custom or semi-custom security enhancements may be added.
  - ➢ SoC Basically fall into four categories
    - Special purpose chips that have degree of programmability
    - ASICs designed with system blocks
    - FPGAs
    - Certain Programmable Processor Arrays (Cypress's PSoC for example)
  - ➢ These are not necessarily secure devices, though as we will see some can be.

System On a Chip ASIC



FPGA SoC Architecture



SoC Characteristics

- All key parts of the system are on a single die and therefore in a single, protectable, package.
- If based on FPGAs, reconfigurability can be used for both updating the system and improving graceful degradation while maintaining a secure processor.
- High performance systems are realizable.
- Depending upon the SoC chosen, and the development methodology, development cost will be moderately higher to significantly higher than for designing with convention, non-secure processors.
- Key management is provided on the device. However often, as in the Virtex IV, the key management hardware is only available to the load module. Other key management is up to the user!

FPGAs as High Performance Secure Processors

“When used in conjunction with the security monitor, NSA found the V4 to be a robust architecture capable of processing classified information and maintaining a very high level of security.” [3]

First Steps

- There are several initial factors to keep in mind when designing a secure processor based on FPGAs:
  - ➢ Select a device that has security features. Either an encrypted PROM for volatile devices or secure version of non-volatile devices.
  - ➢ Select the key storage method: Static or Dynamic?
    - Static keys allow the device to be programmed prior to board installation.
    - Static key are rarely (if ever) erasable.
    - Volatile keys are (usually) more easily erased.
    - Volatile keys *theoretically* reduce reliability due to the need for either continuous power or an external battery backup.
  - ➢ Perform the design using a secure methodology. For example, when encrypting the bit stream for a Virtex IV using the ISE 8 tools the encryption key is copied into the design reports. Thus if one is not careful, the key could be released as part of the documentation package.

Design Methodology

- Design the application for security. This can be trickier than it sounds.
    - Isolation: Internal visibility due to I/O pins or cross coupling.
    - Controllable debug access
    - Device architecture
    - Design software: the design software must support these features (this usually requires additional tool support.)
    - Power supplies
    - Shielding
    - Board layout

Design the Application for Security

- Are there security monitors available for the design?
  - What requirements do these place on the design?
  - How secure is the IP?
- If the design is going to use other IP (usually the case for modern designs) do they expose the system to security threats?
- Consider security requirements from the start, and as the design progress:
  - SPA/DPA: Board layout, filters, operational sceanarions, etc.
  - DEMA: Board lay, enclosures, etc.
  - Isolation between data streams
  - Isolation between execution Streams

Design the Application for Security (continued)

- Make sure the tools don't "optimize" out these efforts!
- If external RAM or PROMs are required for code execution, data contents, storage, etc, encryption can be added to the FPGA fabric just as discussed in the section on secure processor architecutres.
- However, keep in mind that this may affect things like DMA operations into Block Ram, Processor Wait States, cacheing, etc.

References

[1] Cryptography for Engineers Who Couldn’t Care Less, Jim Turley, Microprocessor Analyst and Editor-in-Chief, Embedded Systems Programming. Rabbit Semiconductor White Paper W106.

[2] *Increasing System Security by Using the DS5250 as a Secure Coprocessor,* Dallas Semiconductor’s Application Note 3294

[3] FPGA-BASED SINGLE CHIP CRYPTOGRAPHIC SOLUTION, Mark McLean, National Security Agency, Jason Moore Xilinx Corporation, MILCOM 2006 paper and presentation.

Acronyms

ASIC	Application Specific Integrated Circuit
ASSP	Application-Specific Standard Product
AT	Anti-Tamper
DEMA	Differential Electromagnetic Analysis
DMA	Direct Memory Access
DPA	Differential Power Analysis
EFP	Environmental Failure Protection
EMA	Electromagnetic Attack
HIPA	The Health Information Protection Act
IA	Information Assurance
IP	Intellectual Property
ITAR	International Traffic in Arms Regulation
NRE	Non-Recurring Engineering
PROM	Programmable Read Only Memory
RAM	Random Access Memory
RE	Reverse (Recurring) Engineering
SoC	System on a Chip
SOX	Sarbanes-Oxley Act of 2002, Public Company Accounting Reform and Investor Protection Act of 2002
SPA	Single (Simple) Power Analysis