

Reasons for using secure microcontrollers

We deal on a daily bases with devices incorporating microcontroller or microprocessor logic. Some of these devices or applications would yield more benefits if the hacker could get his hands on the code running inside the micro. Thus, “secure microcontrollers” appeared on the market dedicated to applications requiring an increased level of protection: secure card readers, gaming technologies, fiscal printers, metering applications, payment terminals, intellectual property protection, cryptography devices etc. These types of micros are considered protected by default against hacking techniques, through smart protective measures taken at a silicon level. It is also very difficult to get legitimate information about them, or even a decently specified datasheet from the manufacturers. Big players in the microcontroller market will not openly share secure microcontroller datasheets on their websites, and they will not easily provide it even on request (you would have to justify a large volume order and be known and trusted in the design field to get your hands on such a document)

Direct attacks

Basically, any unsecured microcontroller may be subjected to direct attacks like code injection. The most known method of code injection is known as “buffer overflow” and its aim is to actually take control of the actual microcontroller itself by forcing it to execute hacker defined software. This method exploits also software weaknesses (not only hardware) by corrupting the stack and forcing the instruction counter to jump to a preferred address location upon return from an interrupt. Such methods of hacking are facilitated even further by the complexity of today’s systems which include also external memories used by microcontrollers. It is relatively easy to physically replace a flash chip or even a RAM module and have the microcontroller do what you want it to do.

Several methods of defense have been developed against buffer overflow attacks. The first one is to inherently limit the memory access within the microcontroller and to define the stack limits. A second protection method, which is rather expensive (but in some security applications price is not always the first criteria), is to equip the micro with a physical shield of thin metallic wires that will encapsulate the critical parts of the system. Access inside the integrated system is not possible without tampering with this mesh, which is permanently monitored by the micro. Any attempt to break the mesh results in a general deletion of all the memories thus preventing access to their content. The third protection method refers to the attempts to sniff and then replace the data over the external memory buses and this is dealt with by encrypting the data transfer between the micro and other external memory chips.

The all-around useful JTAG interface is actually a back door to the internals of the microcontroller, exactly due to the purpose it has been made for. To a secure microcontroller, however, this is an unwanted weakness and it has to be dealt with. Manufacturers simply decided to remove it from the secure micros, but in order to still give the developers the option of fast programming and debugging they can generally release two versions of one secure microcontroller: one equipped with JTAG and dedicated to

development, one without JTAG and dedicated to deployment in the end system.

SPA (Simple Power Analysis)/DPA (Differential Power Analysis)

A more “hardware” approach to spying a micro is through power analysis. This has the advantage of being a non-invasive method, meaning no components have to be unsoldered and no lids have to be cut open. The truth is that any logic device leaks information about the data it processes momentarily. This info can be snatched either by analyzing the radio emissions of the micro (which is rather expensive and has to be done in an EMC lab with dedicated equipment) or by analyzing the current consumption of the device (rather cheap and achievable with just “above the average” lab equipment). Every executed instruction generates a specific current consumption and therefore it can be determined. At an extreme, even the data transferred between the CPU and memory may be guessed based on this consumption which is a real threat even to micros using encryption and other algorithms, because at the moment of transfer between memory and processor, the keys for these algorithms are exposed. These attack methods are known as Simple Power Analysis or Differential Power Analysis, based on some detailed differences between them.

Fortunately, even if these types of attack are dangerous, a broad range of mechanisms may be employed to protect against them. Secure micros are built so that they randomly execute dummy instructions, fake additional current consumption, falsely jitter the clock, randomly trigger interrupts etc. Out of all these, the most effective is the scrambling of the current consumption, as it is not related to the real activity of the CPU anymore and thus it does not provide any valuable information to the attacker.

Side channel and fault injection attacks

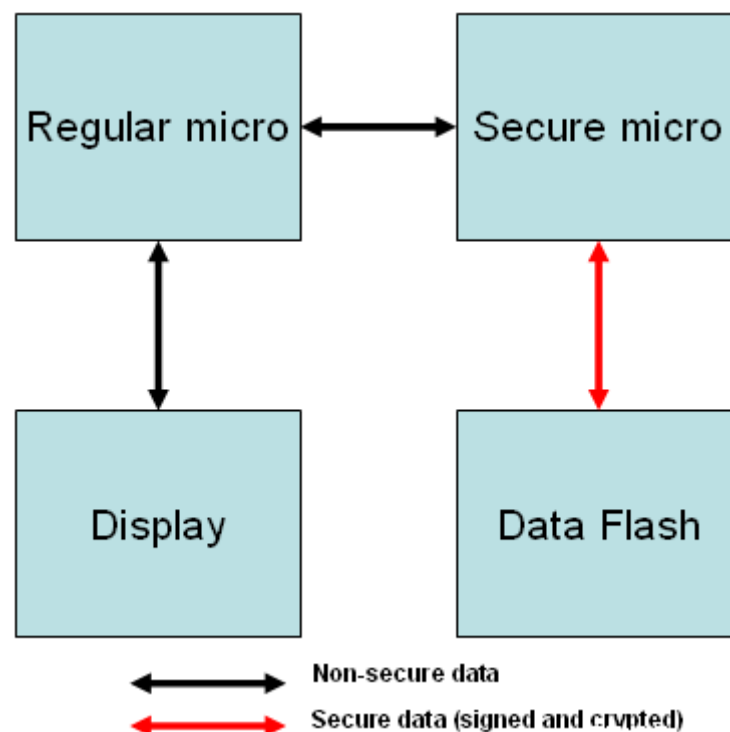
These types of attacks may be both intrusive and non-intrusive. Their aim is to disturb regular behavior of the CPU resulting in various advantages for the attacker. The micro can be made to enter test mode (where full access to the program memory is granted) or the configuration bits can be ignored (out of which some specifically refer to data protection). Side channel attacks generally consist in glitches applied either on some pins of the microcontroller or at a silicon level.

We will describe two situations. The first situation refers to an application where the microcontroller is used to read a PIN access number, to compare it and then activate the opening of a door. A power analysis method (SPA/DPA) may be carried out in order to determine when the micro is actually executing the comparison instruction, verifying the PIN entered from the keyboard against the PIN stored in the internal memory. A glitch applied on the proper pin of the micro at this moment may determine the CPU to skip the comparison instruction (and thus jump directly to granting door access); or it might corrupt a counter value thus giving unlimited tries to guess the PIN by entering successive values from the keyboard (such a system generally grants 3 trials of entering the PIN before completely locking the access to the door; the number of trials is stored as a simple value in a memory location, and if this memory location can be corrupted from 0x03 to 0xFF, for instance, then you get 256 trials at guessing a 4

digit PIN).

The second situation describes a micro which has regular program memory protection bits/fuses and which allows or does not allow access to the memory again, based on a comparison instruction between the configuration location and a hardwired value. A glitch attack can determine a false execution of the comparison instruction, or a false jump, thus rendering the correct protection setting of the configuration bits useless. Such glitch attacks are sometimes more effective if they are applied at silicon level, rather than on the pin of a microcontroller, but this involves taking out the die from its package, which is an invasive and more expensive method of attack. Sometimes it is more useful to subject the die to some UV light rather than applying voltage glitches, but all the time the purpose is to inject a fault in the code execution.

The methods employed for protection against such attacks range from placing light detectors in the silicon (to detect UV light) and scrambling the silicon layout (this is NOT preferred due to affecting performance) to placing a very thin grid of metallic wires over the silicon which detect any attempt to probe the die. Applications of secure micros Once the secure microcontroller appeared on the market, a new broad range of applications were developed, which were previously prone to IP theft. Many applications deal with both sensitive and non sensitive data, and these can employ a double processor architecture, where one general purpose processor deals with display and other non critical secure functions, while a secure microcontroller takes care of distributing security data (signed and encrypted) to and from a memory.



Such architecture has the advantage of preventing any attacks on the memory itself and of preventing any unwanted retrieval of secret information. It can be used for gaming and vending machines, fiscal printers, etc... Other applications for secure micros may be on-the-fly encryption of data transferred between two machines, network access control based on tokens (most commonly known for the small USB sticks sometimes distributed in corporate environments and which are referred to as "tokens"), metering applications (where billing information and consumption data has to be transferred between meters installed at the customer site and servers installed at the provider site), payment terminals etc.

However, the first thing that comes into mind when talking about secure micros is IP protection. Many devices have a simple hardware that can easily be cloned, and in order to prevent the cloning of the entire system a secure micro needs to be used for protecting the software residing in the program memory. Thousands of copy-cats would have been prevented by the use of such devices, and this is probably the main reasons for which the market for secure micros has been developed.

Even so, it is not a huge market and the competition is not very tough. Only a few players venture in this area and the most known are Atmel, Maxim and ST. Out of these, Atmel seems to be the one most willing to give ahead information, and even if we are not talking about real datasheets or instruction sets it is still more than the others do. Sometimes such info can be enough in order to tip the balance in their favor when deciding on a supplier. It is the duty of the project manager to convince the microcontroller manufacturer that it is a real and worthy project, because as I mentioned above, only trusted clients are being sold these kinds of devices.

rmashmoul