

Embedded in Thin Slices

Internet of Things Security (Part 3)

Secure Processors

```
if ($_GET['defaults'] == 1)
{
    $reboot_needed = 1;
    $response = `/apps/cmdxmlin defaults`;
}
```

In this next part of his series on IoT security, Bob looks at the security features of a specific series of microprocessors: Microchip's SAMA5D2. He examines these security features and discusses what protection they provide.

By
Bob Japenga

A couple of years ago, I asked if you, the reader, would be interested in knowing more about the specific security features of some of the new offerings on the market. In my last article, I mentioned a specific family of microprocessor units (MPU) that we have used at our company. The entire article concentrated on one of the security features the MPU contained (features preventing specific side channel attacks) that I was not familiar with. This month I want to step back and look at each of the other specific features and describe what they are and when we need this kind of protection. There are many MPUs from other vendors that provide this. Therefore, this article is less about the Microchip (formerly Atmel) SAMA5D2 and more about helping you understand what each of these security features will bring to your design.

SAMA5D2 BASICS

Before we delve into the specific security features of this chip, let me give you a little background on the it. In 2008 our company created a line of products for a client built around the Microchip AT91SAM9G20. Overall, it worked well for us. The part was well supported in Linux and that was important for us. However, we had significant electromagnetic interference (EMI) issues with the chip that were very costly for us to resolve. The data sheet had a little one-liner that warned us about this but we missed

the significance of that warning amidst the thousands of pages of documentation. In 2017, we were ready to start a new project for this client and did not think the existing processor had enough life in it for the next 10 years of production. After reviewing a number of candidates, we settled on the SAMA5D2 (**Photo 1**). It too appears to have a solid Linux community around it. It was a low-power MPU that could run at 500 MHz and included a built in floating point unit (FPU). It seemed like a good fit for our purposes.

Although we didn't select this chip for these features, the SAMA5D2 family includes a wide range of security features. The datasheet mentions these: 5 KB of internal non-imprinting scrambled SRAM (1 KB non-erasable on tamper detection and 4 KB erasable on tamper detection); 256 bits of scrambled and erasable registers; Up to eight tamper pins for static or dynamic intrusion detections; environmental monitors for temperature, voltage, frequency and an active die shield; Secure Boot Loader; On-the-fly AES encryption/decryption on DDR and QSPI memories (AESB); and a real-time clock (RTC) with time-stamping on security intrusions. Let's take a look at each one (of course only in thin slices).

NON-IMPRINTING SCRAMBLED SRAM

Simply stated, a small amount of memory (5 KB) is set aside for sensitive data like passwords and encryption keys. 4 KB can be automatically and permanently erased upon

tampering and the other 1 KB not erased following a security intrusion. Common to all of our IoT designs is the encryption and decryption of data using private keys. Embedded IoT designs present different challenges for key management than do smart phones and laptops. Embedded devices are deployed for many years. Usually a single server interfaces with tens of thousands of devices. One of the challenges facing us as designers is where to store the keys and passwords. We use either symmetrical or asymmetrical encryption using the standard AES encryption. Sometimes we use asymmetrical encryption which uses a public and private key as illustrated in **Figure 1**. On the *Circuit Cellar* article materials webpage, there's a link to a YouTube video about how asymmetrical encryption works. At the end of the video, they show us placing the private key in a safe. Looks good for the video. But where is the safe in our embedded system? The non-imprinting scrambled SRAM is one such safe. We design systems that contain passwords and private keys that if disclosed could compromise an entire fleet. With cyber terrorism a major threat to even small players like us, protecting this sensitive data is paramount.

You might say: "But my keys are stored in non-volatile memory in the application." Yes, but when the application program runs in Linux, the application is usually running in volatile memory. Or you might say: "I generate the private key every power up." In both cases, the private key and password ends up in volatile memory. Okay, then you might say: "But how can someone read the contents of my keys and password stored in volatile memory?" One method plays upon a susceptibility that conventional SRAM has to imprinting. With SRAM, the way that the data is stored imposes stresses on the cell's transistors that can create a long-term constant bias voltage on the cell. Over time, the state of the charge of the cell—a one or a zero—can be read out long after the power has been removed from the system. When heat is applied to the chip, the probability of imprinting increases. So, without imprint protection, your cyber attacker has tools to read your passwords and private keys long after he or she has removed power from the device. The SAMA5D2 has the ability to permanently erase these passwords upon power removal and upon tampering—more about tampering later.

In addition, the data in the SRAM is scrambled with a 32-bit key. In other words, if we don't stop the attacker at the moat and they are able to read the key, we have the boiling oil to pour on the attacker because the key is encrypted.

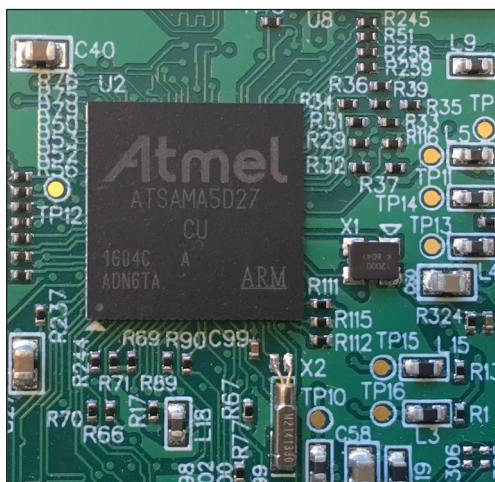


PHOTO 1

The Microchip SAMA5D2 is a low-power MPU family includes a wide range of security features.

ANTI-TAMPER PINS

Many years ago, I ran a youth group at our church. An annual event that generated a lot of interest was the great road race. The students would break up into teams of three or four and receive a clue and a panic envelope at each station. Their job was to decipher the clue which told them where the next station was. Then they would travel to the next station where they would get the next clue. Should they be unable to decipher one of the clues in the allotted time, they could open the panic envelope which would provide the next location. At the end of the race, the first one with the fewest opened panic envelopes won. One year one team zipped through the course. No panic envelopes were opened. I suspected fraud but couldn't prove it. The next year I instituted tamper detection on the envelopes. The same team finished first with

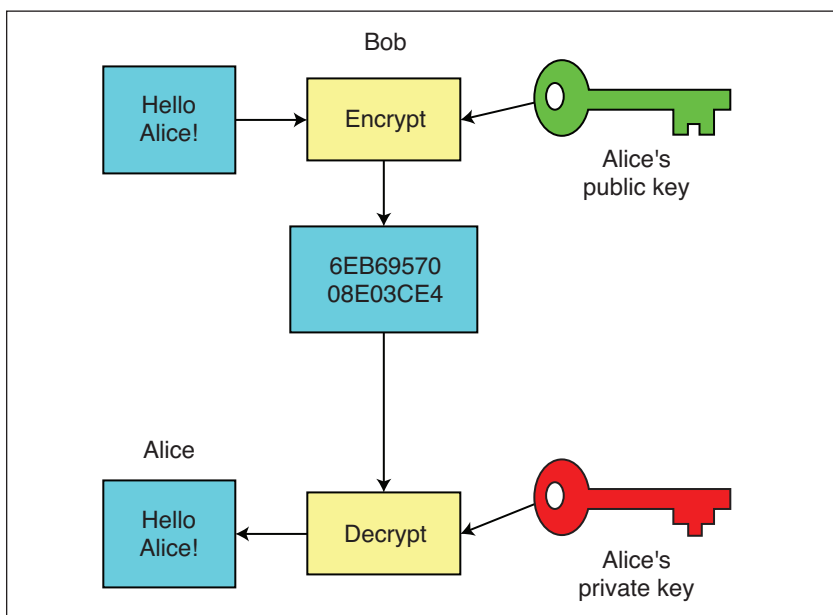
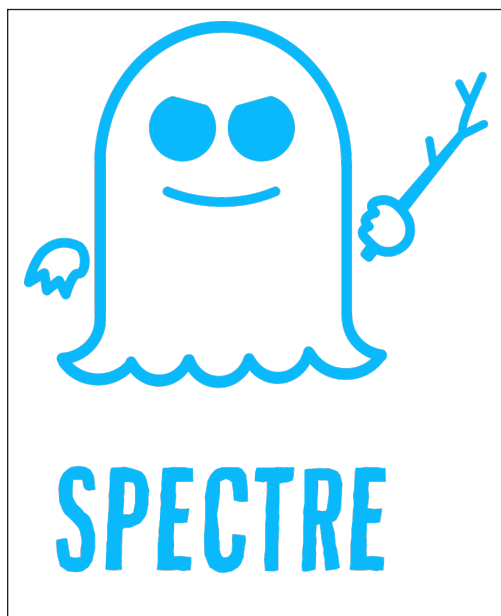


FIGURE 1

Asymmetrical encryption uses a public and private key as illustrated here.

FIGURE 2

System designs that make use of deep instruction pipelines with branch prediction are vulnerable to a Spectre-like attack.



no unopened envelopes. But they had all been tampered with and they were disqualified. This provided great life lesson material for our next meeting.

Our embedded IoT devices have much more at stake. Ransom costs and loss of reputation from a disclosed security breach could sink a small to medium size company. The SAMA5D2 provides 8 pins used to detect tampering. The details of how it is done is not important for this article and not available without an NDA with Microchip. But why do we need tamper detection? Isn't it obvious? Your device has vital data that you want to protect inside it. And if you deploy enough of these devices you could be vulnerable to a ransom attack or worse. Tamper resistance is not new. For many years we have created secure devices with tamper detection. Even our old dome electric meters had tamper detectors. In one project we epoxied a device that contained postage and provided some kind of detection that an unauthorized entry had happened. With the SAMA5D2, you not only get notified of the tamper attack, you also have the option to automatically erase the sensitive data like your private keys.

Recently there has been a lot of hype about the Meltdown and Spectre security bugs. One reputable site has stated that:

Spectre affects Intel, AMD and ARM processors, broadening its reach to include

mobile phones, embedded devices, and pretty much anything with a chip in it. Which, of course, is everything from thermostats to baby monitors now.

Although the article's broad brush covered "anything with a chip in it," the article assumes that these embedded processors have branch prediction. So, ignoring the hype—since not anything with a chip in it has an MPU with branch prediction—we have to admit that most of our new designs do have deep instruction pipelines with branch prediction. Therefore, by their architecture they are vulnerable to a Spectre like attack. With the advent of these techniques for extracting secret keys from our devices, we need to beef up our layers of protection. Unlike a PC, most of our embedded systems would require some kind of physical access to be susceptible to the Spectre method of extracting our sensitive data. So, if we have sensitive data that needs protection and a cyber terrorist can get hold of one of your devices, we need tampering detection. If the feature is easy to use, why not protect your systems against such intrusions?

One nice feature of the SAMA5D2 is that it provides a RTC that provides a time stamp and event counter for tamper detection. That could prove useful in identifying the extent and timing of such attacks. Also—although not defined in the public data—you can imagine how the die shield feature could also help prevent some cyber terrorist from taking the lid off the part for additional nefarious activities including some of the side channel attacks we talked about in my April article (*Circuit Cellar* 333).

Some of the imprinting attacks require both out-of-spec temperature, frequency and voltage conditions. Again, the public data sheet doesn't provide a lot of information about how these work, but you can imagine that locking the safe or erasing the contents of sensitive data could be a useful deterrent against such attacks.

Secure Boot-Loader: One of the critical features needed in all of our IoT devices is the ability to update the software remotely once it is deployed. This brings with it two separate problems: How do we prevent someone from reading the code as it is being transmitted from the server and thus be able to reverse engineer all of our sensitive data? And, how to make sure that the code is coming from a trusted source? Encryption and authentication are critical for that. In all of our IoT systems we provide both of these features. But once the program is decrypted and authenticated it is usually stored in an on-board memory. We usually avoid storing the program on an

For detailed article references and additional resources go to:
www.circuitcellar.com/article-materials

RESOURCES


Microchip Technology | www.microchip.com

SD card because it is so easy to reverse engineer. But unencrypted on-board memory is now equally easy to reverse engineer.

A secure boot-loader allows you to boot from encrypted and signed images from either an SD card or on-board memory. This prevents someone from removing the memory chip and reverse engineering your code since all of the code is encrypted. Remember: What if someone motivated to extract money from you got all of your code by just removing the chip from your bank card? A secure boot-loader allows us to keep the code that was encrypted over the air to be stored in encrypted memory.

On-the-fly AES Encryption of Memory Access: As part of the Secure Boot-Loader are the secure locations for storing AES keys used in the Secure Boot-Loader. The SAMA5D2 provides the ability to encrypt not just the bootable file, but the DRAM where the Linux apps run. This provides additional protection from the intruder who would set up a logic analyzer on the data bus to reverse engineer your design. A lot of these features can be performed in software. Having this feature in hardware provides just another layer of security against sophisticated terrorists out to extort money and reputation from your company or your client's company.

CONCLUSION

It is a wild and wooly world out there. There are people in this world who are desperate for either making a name for themselves or making you pay a lot of money to re-instate the fleet of IoT devices you deployed. The SAMA5D2 MPU provides a number of built in features to help us create more bullet-proof IoT devices. We have looked at each one—but of course—only in thin slices. 

ABOUT THE AUTHOR

Bob Japenga has been designing embedded systems since 1973. In 1988, along with his best friend, he started MicroTools, which specializes in creating a variety of real-time embedded systems. MicroTools has a combined embedded systems experience base of more than 200 years. They love to tackle impossible problems together. Bob has been awarded 11 patents in many areas of embedded systems and motion control. You can reach him at rjapenga@microtoolsinc.com.

