Battery Authentication for Portable Power-Supply Systems

By
Ken Dietz
Security, Microcontroller and Technology Development Division
Microchip Technology Inc.
Chandler, AZ
www.microchip.com

Introduction

Battery authentication for portable power-supply systems is a growing trend, in part due to the recent increase in personal injuries related to the improper charge profile of counterfeit rechargeable batteries. Additionally, ensuring the marketing of safe, rechargeable battery-powered systems is a growing concern for many government agencies, so electronics manufacturers must now ensure that their products will not accept incompatible batteries.

As the need for battery authentication continues to grow, portable-system designers face a number of questions related to the inclusion of battery authentication in their systems. Designers are challenged to decide when systems are secure enough for certain applications. Where is the line drawn to say that one solution is better than another, or that a given solution should be implemented in hardware as opposed to software? Answers to these questions are not as clear as one might expect. Software code is designed using a variety of compilers and assemblers, but the compiled software is stored in hardware at the program-memory level. At this level, the hardware reads the instructions and completes tasks. Whether the solution is hardware- or software-based, at some point in time, the two must connect.

To complicate things further, the topic of battery authentication deals with digital security and encryption, which is a very specialized field with several pitfalls. While most people want a system that is 100% foolproof, the reality is that security is a trade-off between cost, complexity, and the value of what is protected. In this case, what is protected is the possibility of the user placing the wrong battery in a portable device, as well as protecting the manufacturer’s brand from being associated with unsafe devices.

Challenge/Response Systems

One of the most common ways to authenticate a battery and ensure that it is from a valid source is with a challenge/response system. Challenge/response authentication circuits, also known as Identify Friend or Foe (IFF) circuits, take a variety of forms. Some details for these types of circuits are discussed below. Understanding them helps system designers defend against attacks.

The basic system is implemented in such a way that one part of the system, the host, issues a challenge to the other part of the system, the token, when the two components begin to communicate. After the challenge is received, the token computes a response and transmits the results back to the host system. The direction of the challenge and response can be reversed or even transmitted in both directions. Additionally, either side of the system can randomly transmit the challenge and response at varying times in order to increase the security of the authentication process.
Before discussing challenge/response architectures in greater detail, it is necessary to clarify a couple of terms. Unidirectional authentication refers to a system that contains at least one host and at least one token, where one of the two challenges the other with a random number and waits for a response. An extension of the unidirectional authentication concept can include a response that is not sent directly back to part of the system that issued the challenge. For example, remote controls transmit and do not wait for a response. Instead, the authentication takes place at the site of the receiver, where some control sequence takes place to turn on or off some type of circuitry following authentication. Hopping-code transmitters and receivers are popular in this field of electronics, and the concept can be easily extended to battery authentication.

Bi-directional authentication refers to a system that implements some sort of authentication sequence between the host and token, where the host authenticates the token with a random challenge, and the token, in turn, authenticates the host with a random challenge. A key point to making a bi-directional authentication system more secure is to design the system so that the token validates the host CPU on the charging system, prior to receiving a challenge. In doing so, the token residing within the battery pack will not respond to an invalid charger.

A typical timeline for a basic challenge/response system is shown in Figure 1. Depending upon the implementation, timelines may include additional or different features not shown here, and some of these features may drop out, altogether. After the host processor issues a random challenge, it should wait for a short, fixed period of time for the response. If the valid response is not received within this fixed window of time, then the host must block the token to prevent charging an incompatible battery. Figure 1 represents unidirectional authentication, where only the host is authenticating the token.
Cryptographic algorithms decode the challenge and calculate the response. There are several algorithms available, and algorithm selection relates back to the point that security is a trade-off between cost, complexity and the value of what is being protected. It is important to note that even the strongest algorithms are worthless unless the remainder of the system is managed in a secure way. When choosing an algorithm, it is usually best to stay with something that has withstood the test of time, since most people do not have the expertise to develop their own algorithm.

Two small block-cipher algorithms that are available for battery-authentication circuits are the eXtended Tiny Encryption Algorithm (XTEA) and the KEELQ® algorithm by Microchip Technology Inc. XTEA is a publicly available block cipher that was developed by Roger Needham and David Wheeler in 1997. It uses a 128-bit key and has a 64-bit cipher, and is known for its compact size and simplicity. XTEA is attractive for processors that primarily use assembly language.

The KEELQ technology is a proprietary algorithm that has been widely used for more than a decade in applications ranging from garage-door openers to passive-keyless-entry systems. The KEELQ algorithm uses a 64-bit key and has a 32-bit cipher. The KEELQ technology is even more compact than XTEA in terms of program memory and RAM requirements, but it does take longer to calculate a solution.
In terms of applying these algorithms to battery authentication, the host processor is trained to know which batteries are intended for use by the charging system. The training can occur at the consumer level, where customers are required to take their portable device into authorized dealers so that the end system “learns” its batteries. Additionally, some manufacturers implement solutions into their handsets to validate batteries. This validation procedure is implemented each time that the handset loses power. Similarly, when customers purchase new cell phones, the phone is programmed with the new phone number and is trained to the carrier network. During this timeframe, the phone’s battery may also be “trained” to the handset and charging system. The detail of the “learning” scheme will depend on the individual system and how transparent to the user the manufacturer wants to make it. Preferably, every battery should be unique, so that if one battery is compromised or cloned, the whole system is not compromised. This may include “blacklisting” certain numbers to prevent batteries with a specific serial number and cryptographic key from being charged.

Authentication Example Using a PIC10F Microcontroller

As long as attackers continue to break in to secure systems, design teams will face the on-going challenge of making systems more and more secure. A good balance of cost-effectiveness, time-to-market and flexibility exists in the form of microcontrollers. By implementing battery-authentication solutions in Flash-based microcontrollers, changes to keys, algorithms and functionality can be made without a hardware redesign. Microcontrollers also allow designers to introduce custom features into their systems, such as battery gauging and safety-related features, such as temperature and current monitoring.

Simple decode circuits, such as that shown in Figure 2, are good starting points for embedded-system security. Systems like this can be a launch pad for solutions with more features and security. The circuit provides some degree of security, in that a challenge is issued and a response is required before charging commences. To make the system even more secure, a bi-directional authentication protocol and multiple decryption keys and/or serial numbers are easily added, as previously described. This type of system uses a pulse-width modulation (PWM) communication scheme to communicate between the host and the token. If a processor with an on-chip analog-to-digital converter is used, then temperature-sensing circuits and low-voltage-monitoring features can be added to the system.
The system in Figure 2 is based upon the PIC10F200 microcontroller on the token side of the circuit, and the PIC16F877A microcontroller on the host side of the system. The token software is written in assembly language, in order to make it fit within the 6-pin PIC10F200 devices. Another reason for using assembly is that the KEELQ technology is proprietary to Microchip Technology and requires the token implementation on a PIC® microcontroller. The PIC16F877A code is written in C to make it portable, since the host controller will vary depending upon the final system. This system is designed using Microchip's KEELQ authentication algorithm. For a download of the application software and complete authentication schematics, please follow the associated links at the end of this article. Figure 3 shows the flow for the token software.
Attacking Integrated Circuits

For battery authentication, the attacks are typically by an entity that wants to clone batteries and sell them as original equipment in the aftermarket. The aim, therefore, is to attack the integrated circuit in the battery and make a functional duplicate.

Generally, there are two types of attacks on security chips -- non-invasive attacks and invasive attacks. Non-invasive attacks include analyzing and characterizing an integrated circuit such that the component packaging is kept intact and the semiconductor is exercised through a number of methods. The brute-force technique stimulates the component by providing all known challenges to obtain all known responses for a given component. Solutions for challenges in brute-force attacks are typically held in RAM or some type of database within the attacking system. Other non-invasive attacks include approaches that involve powering the device under test into strange modes that will cause the chip to begin functioning in unusual states, or by attempting to find test modes to gain access to the program memory.
On the other hand, invasive attacks are implemented by removing the packaging of a semiconductor with some chemical or mechanical methodology. The chip is then probed or modified at the silicon level in order to gain access to stored information. Additionally, invasive attacks are implemented by inducing some physical phenomenon onto the device in order to reveal the code within the device, such as changing the configuration fuses, modifying transistor properties, or even reverse engineering the integrated circuit.

Defending Integrated Circuits

Defending against brute-force attacks relates directly to the size of the cipher in the cryptographic algorithm and the response time of the system that received the challenge. Consider a cipher size of 32-bits. An attacker must store or calculate nearly 4.3 billion combinational solutions for this cipher size. Considering the fact that it typically takes about 100 ms to complete a challenge/response authentication sequence, a brute-force attack will take close to 817 years to search for a guaranteed solution. However, if the attacker gets incredibly lucky and guesses the first answer, then the next solution will be available in very short order. As a result, system designers typically agree that a 32-bit cipher or 64-bit cipher is sufficiently large enough to meet the needs of battery-authentication systems.

Defending against invasive attacks is a factor for all semiconductor manufacturers. This field is constantly evolving, as semiconductor manufacturers make their processes more secure by reducing their geometry sizes, and make it more difficult for attackers to gain viable information from the silicon and metal levels within the devices. System designers must ensure that the associated variables for a cryptographic algorithm are not all held in RAM at the same time. Instead, blocks of data must be handled at different times in order to make the chip more difficult to probe in real time, rather than making all the information available simultaneously in RAM.

Key Management

Key management is the process of creating, storing and using keys for secure systems. Beyond any doubt, maintaining the security of the cryptographic keys generated for a company’s product portfolio are at the heart of all secure systems. The security is with respect to these values in terms of the application level, as well as at the long-term storage level. If attackers gain access to a manufacturer’s key or to the databases containing the encryption keys, then all of the systems for a given product line will be compromised. During the key-generation process, many manufacturers task at least two individuals to generate encryption and decryption keys. The data is entered by each person separately, with no knowledge of the information that is entered by the other people involved, so that no single individual gains access to the entire key combination for the company’s products.

In the system described here, the cryptographic keys are developed so that they are unique to each device’s serial number. In order to do this, the serial number is combined with the manufacturer key or a random seed value through binary scrambling. With multiple encryption keys, the additional keys are also combined using binary scrambling and are accessed separately within the system at run time. After the keys and serial number are combined, a method such as the use of an encryption or decryption algorithm on the key/serial number combinations scrambles the numbers to generate the final cryptographic key.
Related to key management, systems can utilize multiple manufacturer codes to generate multiple cryptographic keys for any one serial number. With this security measure, the key accessed for the challenge/response process is randomly chosen at the time that the challenge is issued. This increases the complexity of system software, but it also increases the system’s security with minimal non-recurring costs.

**Conclusion**

Battery authentication for portable power-supply systems can be effectively implemented using small microcontrollers in battery packs that communicate with host processors. These types of microcontrollers combine the benefits of small footprint and flexible upgradeability at low costs. Effective key management and the use of a cryptographic algorithm, such as XTEA or KEEOQ technology, add additional layers of security to authentication circuits and, when combined with the tools previously discussed, enable designers to develop low-cost, highly secure battery-authentication systems.

In conclusion, there are three basic fundamentals to battery authentication:

- Make breaking the system more expensive than the value of what the system is protecting.
- Assume that the system will never be foolproof; so build flexibility into it for future upgrades and changes.
- When determining a battery-authentication strategy, consider the whole system, including the design, the components, the manufacturing, and the people who design and manufacture the system. For example, it does not help to have the best cryptographic solution if a single individual knows the critical manufacturer’s keys. In this case, all of the security in the algorithm does not protect a company from attempts to bribe that specific person.

**For more information, please visit:**

1. Microchip Technology Inc. Application Note #827, “Using KEEOQ Technology to Validate System Compatibility:” [www.microchip.com/KeeLoqIFF_AppNote](http://www.microchip.com/KeeLoqIFF_AppNote)

**References:**

2. Delport, V., (July 2005), various interview contributions and for Figure 2. Microchip Technology Inc., 2355 W. Chandler Blvd., Chandler, AZ 85224.

*The Microchip name and logo, the Microchip logo, KEEOQ and PIC are registered trademarks of Microchip Technology Incorporated in the U.S.A. and other countries. All other trademarks mentioned herein are property of their respective companies.*