Application Note 150
Small Message Encryption using SHA Devices

www.maxim-ic.com

INTRODUCTION

When systems communicate telemetry or control information between peers the security and authenticity of the communicated data may be important. If the medium is public or could be subject to compromise, securing the communications path becomes an issue. But encrypting control and status messages that are passed between subsystems on networks, telephone lines, or RF channels usually requires extensive microcontroller resources, and the maintenance of secrets (keys) is usually the weak point in the system. Changing or customizing critical system secrets is often impossible in ROM-based equipment, which further reduces the security of the system.

Dallas Semiconductor manufactures low cost 1-Wire® memory devices that contain fast, powerful cryptographic engines. Some of these devices have the ability to perform the SHA-1 hash very quickly, and to securely store, protect, and rotate secrets. These devices can be used with small microcontrollers and limited resources to provide strong small-message encryption and peer-to-peer authentication between subsystems.

STRONG CRYPTOGRAPHY WITH WEAK MICROCONTROLLERS

When a subsystem component has limited processing power and memory, advanced cryptography is usually not possible. Secure exchange of data and peer-authentication requires secrets, and microcontrollers are not very good at keeping secrets from clever hardware and software attacks. The solution is to move the cryptographic task to a device that is specially designed to perform these tasks well. This paper will discuss the application of 1-Wire devices that perform SHA-1 hash functions to provide a low-cost, low-overhead cryptographic solution for small message encryption and authentication.

To keep microcontroller code space to a minimum, we will use a simple fundamental cryptographic concept called the one time pad. Given an array of bytes that comprise a message, it can be said that the byte-wise XOR (Exclusive-OR) result of that array of bytes against an array of random bytes results in an array of bytes that are equally random. In other words, no information about the message remains in the resulting array. To put it another way, a byte when XOR'ed against a random byte results in an equally random byte.

If one was to generate an array of truly random bytes (called a pad), one could then XOR that array against any valid message and the result would be an encrypted message that is unbreakable by any cryptographic means (so long as the pad is held secret and known only to the valid participants in the conversation). This result, when XORed again against the same pad, will be restored to the original message. This is a basic tenet of cryptography—the function is simple (XOR) and the power is entirely in the quality and security of the pad, not in the algorithm by which it is applied to the message.

This would seem to be a very simple, very powerful way to perform message encryption, but there are caveats:

1) The pad must be passed from the sender to the recipient in a way that it cannot be compromised. Both parties to the conversation must share the same pad.
2) If the pad is used on more than one message, its strength is greatly diminished—if not wholly compromised. A new pad must be created for each message.

What is needed is the ability to generate an array of bytes (a pad) at will, and then to pass some key with the message that can be used by the recipient to regenerate the same pad. This means that each legitimate participant in the conversation must hold some secret that allows the pad to be regenerated given the key, and the secret must be protected at all costs.

In the world of cryptography, the pad generator that we have described is a function called a one-way hash. This function takes input data and generates a digest from it. This digest is entirely affected by every bit of the input data, and yet is derived in such a way that the input data cannot be discovered given the algorithm and the digest. The best-known and most secure one-way hash function is SHA-1 (secure hash algorithm).

The Dallas DS1963S SHA iButton® performs all the required tasks to serve as specialized cryptographic co-processors:

1) Inexpensive
2) Easily connected to a microcontroller using only one I/O pin
3) Can hold a secret in nonvolatile storage and protect it from attack
4) Can rotate (iterate) the secret easily and with complete security
5) Can quickly generate a cryptographically sound pad using SHA-1

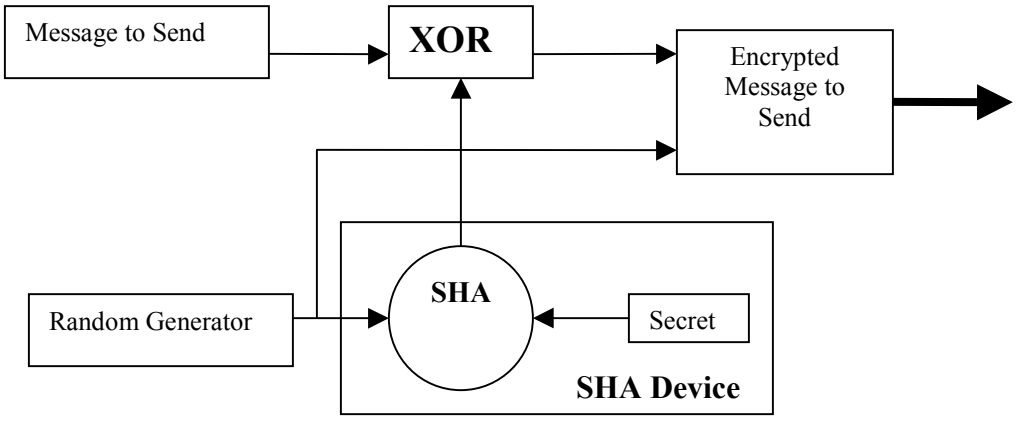
HARDWARE CONFIGURATION

The SHA device requires only a single microcontroller port pin and a pullup resistor. Code in the microcontroller generates the appropriate waveforms to perform two-way communication with the device at either 14KBPS or 140KBPS data rates. Each device includes a globally-unique serial number. The device has the ability to hold and protect secrets and to perform the SHA-1 hash algorithm very quickly.

THE METHOD

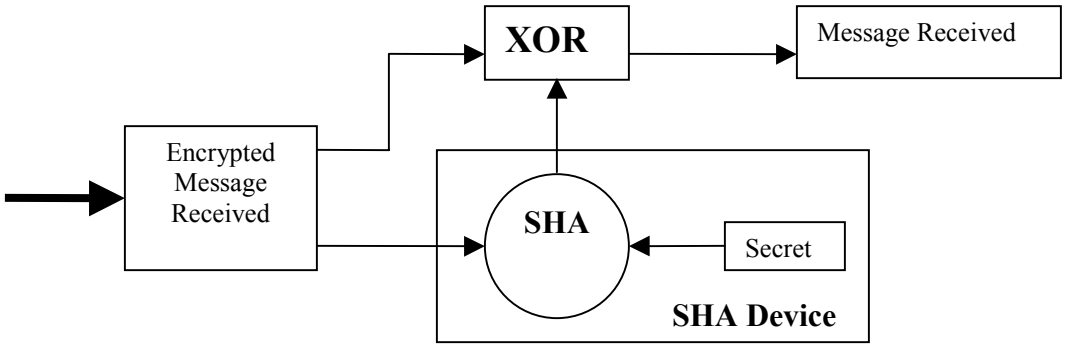
To use the SHA device for peer-to-peer small message encryption, the following simple algorithm is employed:

1) The microcontroller generates a random number and sends it to the device.
2) The microcontroller directs the device to generate a SHA-1 digest using the random number and the secret.
3) The microcontroller reads the 160-bit digest from the device.
4) The microcontroller XORs each byte of the message with a byte of the digest (the pad) to obtain the encrypted message.
5) The microcontroller concatenates the random number and the encrypted message and transmits the result to the peer.



When an encrypted message arrives, the following algorithm is employed:

1) The random number portion of the message is sent to the device.
2) The microcontroller directs the device to generate a SHA-1 digest using the random number and the secret.
3) The microcontroller reads the 160-bit digest from the device.
4) The microcontroller XORs each byte of the message with a byte of the digest (the pad) to obtain the original message.
5) The microcontroller processes the decrypted message.



Despite the simple appearance of this algorithm, it is quite secure. The microcontroller needs only perform basic 1-Wire communications with the device and then XOR the SHA-1 digest, byte for byte, against the message data. Security is assured by the strength of the SHA-1 function. Because the SHA-1 hash function is not reversible, the secret cannot be derived from the message traffic. Without the secret, there is no way to decipher or falsify a message. The random seed value used with each message makes every message unique, and makes the deciphering messages all but impossible.

LONGER MESSAGES

The SHA-1 hash function provides a 160-bit (20-byte) result. When the message to be encrypted exceeds this length, the system may simply perform another SHA operation (to obtain another 20 bytes of pad data).

REPLAY ATTACKS

This encryption scheme provides authentication of the source of each message (because only a valid system participant could have generated a valid message) as well as message data security. However, replay attacks (where a previous valid message is captured and sent again at a later time) are still possible. Several simple methods can protect against this. One method is to include a counter in each message that increments when the message is sent, and then program the recipients to reject duplicate messages. Another method is to have the recipient send a random number (called a challenge) that the sender then includes in the encrypted message. Because the random challenge is very likely not the same for any two messages, a replayed message will be rejected.

RANDOM NUMBERS

The system described herein relies on the cryptographic quality of the random numbers generated by the microcontrollers. For more information, see *Application Note 152, iButton Secrets and Challenges*, <http://dbserv.maxim-ic.com/appnotes.cfm?appnote_number=835>.

1-WIRE SERIAL COMMUNICATION

Communication with the device is done using a single port pin and a well-defined communications protocol. For details on the Dallas 1-Wire serial protocol, see applications information that can be found at www.iButton.com. Generating the 1-Wire waveforms requires simple subroutines that can usually be implemented in a few dozen lines of assembler code in most simple microcontrollers.

Encryption or decryption of a small (<160-bit) message can usually be performed in less than 10ms at slow communication speeds, and in less than 2ms at high communication speeds.

THE SHA DEVICE

There are two forms of SHA device available, each with different characteristics. The DS1963S is a nonvolatile memory with an SHA-1 engine, plus a lithium power source, inside a stainless steel container called an iButton. This device has sixteen 32-byte pages of memory and eight separate SHA-1 secrets. The DS1963S is used in a coprocessor mode to perform the type of SHA-1 function required for small message encryption.

SECRET ROTATION

The SHA devices also provide internal mechanisms to perform secret rotation. The system may send a rotation message to the device and ask that the message be hashed against the old secret to generate a new secret. The device microcontroller does not need to know the old secret to generate the new secret, and the new secret is never revealed—never visible outside the device. In this manner, the system-wide secret can be easily changed (rotated). An attacker is required to have access to the original secret and the rotation message to figure out the new secret. This allows a system to rotate secrets from time to time to assure secret security.

This mechanism also allows system secrets to be installed in multiple pieces and by different parties so that no one party knows everything required to generate the system secret. This secret sharing method further enhances system security.

CONCLUSIONS

The DS1963S SHA iButton can be used with small microcontrollers to provide strong encryption and authentication of control and status messages, telemetry, or sensitive process control information. For low cost and low overhead, it provides nonvolatile memory, secure secret storage, secret sharing and rotation, fast SHA-1 pad generation, and a globally unique serial number. A simple microcontroller needs only provide a single port pin and a few dozen lines of code to attain quality cryptographic security.