Protecting Your R&D Investment with Secure Authentication

White Paper

maxim integrated™

Keywords: SHA-1, SHA-2, SHA-256, SHA256, secure hash algorithm, secure authenticator, secure memory, 1-Wire, authentication

APPLICATION NOTE 3675

Protecting Your R&D Investment with Secure Authentication

By: Scott Jones
Oct 19, 2012

Abstract: In the age of identity theft and faked IDs, achieving positive identification is of paramount importance. This is not only true for individuals, but also applies to electronic products. Manufacturers of nearly all equipment types need to protect their products against the counterfeit components that aftermarket companies will attempt to introduce into the OEM supply chain. Secure authentication provides a strong electronic solution to address this threat as well as additional useful end-product features.

This application note explains the concept of authentication and specifically Maxim's solution in the form of secure authenticators to solve application requirements including intellectual property protection, embedded HW/SW license management, secure soft-feature and status setting, and tamper-proof data storage.

What is Authentication?

Authentication is a process with the objective to establish proof of identity between two or more entities. In the case of one-way authentication, just one party is involved proving its identity to another. With two-way authentication, both parties prove their identity to each other. The most commonly used method of authentication is the password. The main problem with passwords is that they are exposed when used, making them vulnerable to spying.

After reviewing the historical use of cryptography, in 1883 the Flemish linguist Auguste Kerckhoffs published his findings in a groundbreaking article on military cryptography. Kerckhoffs argued that instead of relying on obscurity, security should depend on the strength of keys, because in the event of a breach, only the keys would need to be replaced, not the whole system.

A proven symmetric key-based authentication method works as shown in **Figure 1**: a secret key and the to-be-authenticated data ("message") are taken as input to compute a message authentication code or MAC. The MAC is then attached to the message and transmitted upon request. The recipient of the message performs the same computation and compares its version of the MAC to the one received with the message. If both MACs match, the message is authentic. A weakness with this basic model,

however, is that a static, intercepted message and MAC can later, or subsequently, be replayed by a nonauthentic sender and be mistaken as authentic.

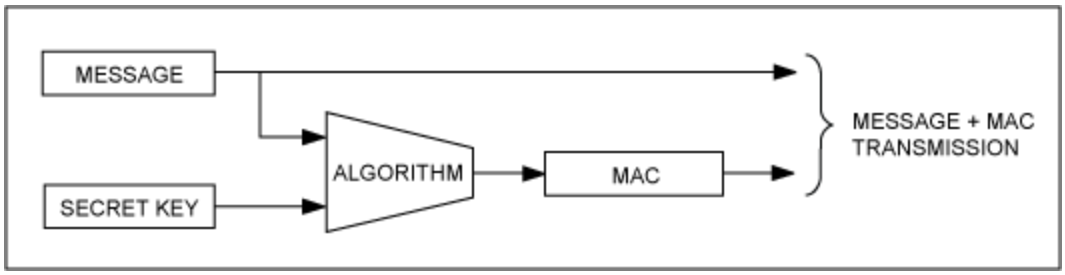

Figure 1. MAC computation model.

To prove the authenticity of the MAC originator (for example, a system accessory), the recipient (i.e., the host system to which the accessory is attached) generates a random number and sends it as a challenge to the originator. The MAC originator must then compute a new MAC based on the secret, the message, and the challenge and send it back to the recipient. If the originator proves capable of generating a valid MAC for any challenge, it is very certain that it knows the secret and therefore can be considered authentic. **Figure 2** shows this challenge-and-response authentication flow and the associated data elements.

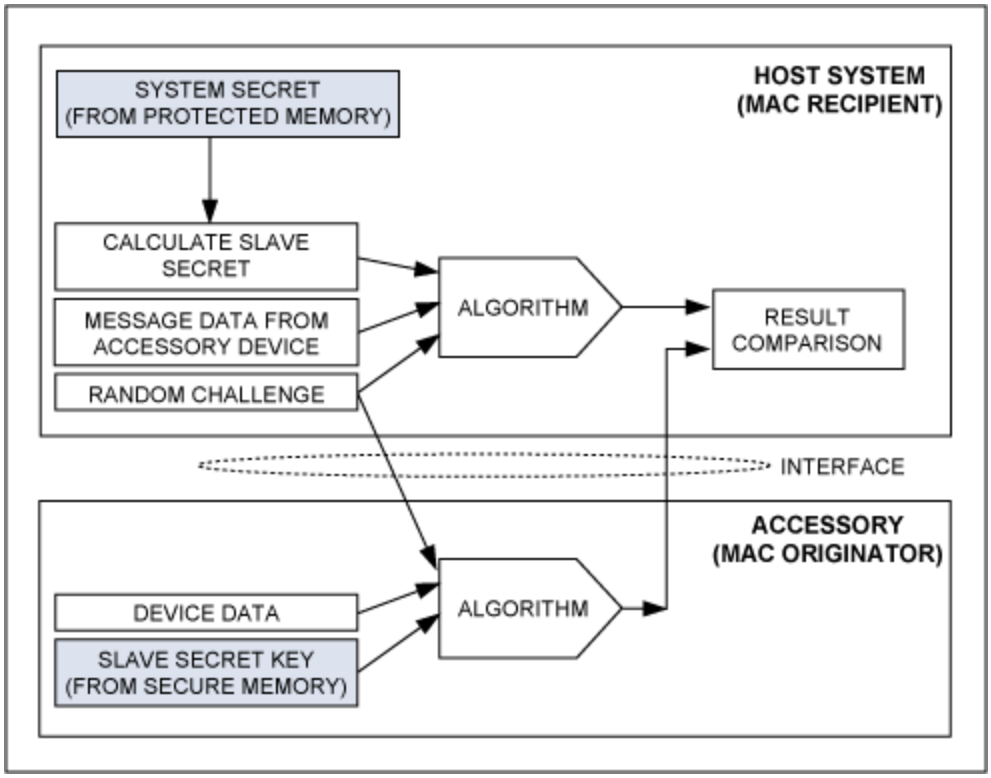

Figure 2. Challenge-and-response authentication data flow.

In cryptography, an algorithm that generates a fixed-length message authentication code from a message is called a one-way hash function. "One-way" indicates that it is mathematically infeasible to conclude from the fixed-length output MAC the usually larger input message, which includes the secret key.

Two thoroughly scrutinized and internationally certified one-way hash algorithms are the FIPS 180 SHA-1 and SHA-2, which were developed by the National Institute of Standards and Technology (NIST). The mathematics behind these algorithms is publicly available through the NIST website. Distinctive characteristics of the algorithms are: 1) irreversibility—it is computationally infeasible to determine the input corresponding to a MAC; 2) collision-resistance—it is impractical to find more than one input message that produces a given MAC; and 3) high avalanche effect—any change in input produces a significant change in the MAC result. For these reasons, as well as the international scrutiny of the algorithms, Maxim selected SHA-1 and SHA-2 for challenge-and-response authentication of its secure authenticators. Further, Maxim implemented the SHA-256 variant of SHA-2 in its latest products.

Low-Cost Secure Authentication—The System Implementation

Thanks to the 1-Wire® interface, a secure authenticator, such as the DeepCover® Secure Authenticator (DS28E15), can easily be added to any system where digital processing capabilities exist, such as to a microcontroller (µC). In the simplest case, all that is needed is one spare microcontroller port pin and a pullup resistor for the 1-Wire line, as shown in **Figure 3**. However, a potential risk with this approach is the use of a nonsecure microcontroller that can be analyzed by an attacker to understand and compromise the security.

Alternatively, as shown in **Figure 4**, an IC, such as the DeepCover Secure Authenticator (DS2465) SHA-256 coprocessor with integrated 1-Wire master interface, can be used to operate and control the DS28E15. While the DS28E15 can be operated with a microcontroller-only approach, the benefits of using a DS2465 include: 1) offloading SHA-256 computation from the host µC; 2) highly secure storage of the system SHA-256 secret; and 3) offloading 1-Wire waveform generation from the host µC.

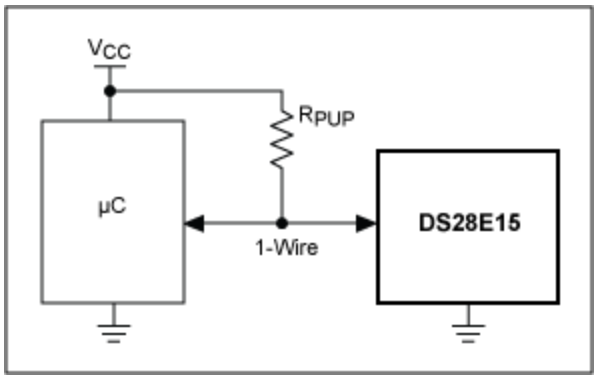

Figure 3. Basic application example.

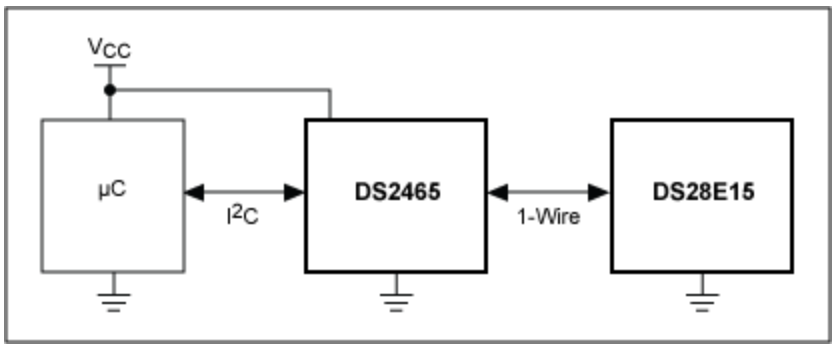

Figure 4. Using a coprocessor to enhance security.

Counterfeit Prevention

Systems with replacement sensors, peripherals, modules, or consumables are commonly targeted by unauthorized aftermarket companies. These counterfeit versions can introduce safety concerns, reduce quality to the application, and in general negatively impact the OEM solution. Introducing secure authentication into the solution enables the host system to test sensor or module authenticity and to take application-specific action if a counterfeit is detected. As shown in **Figure 5**, a challenge-and-response sequence between the system and attached peripheral is exercised to confirm authenticity.

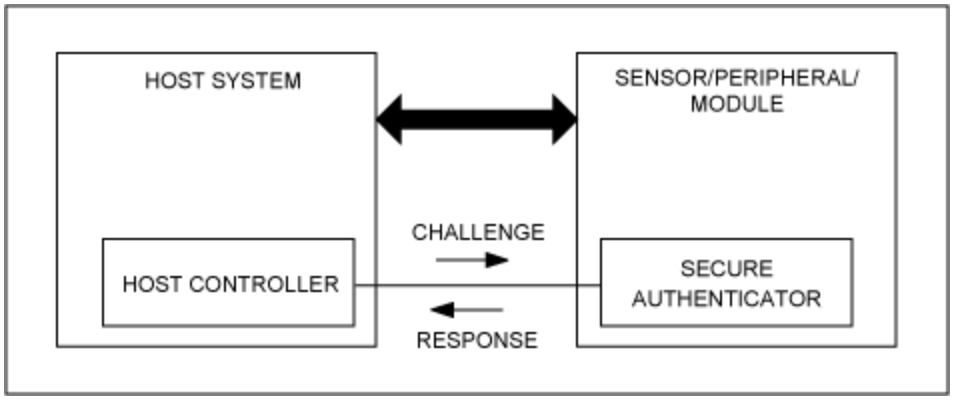

Figure 5. Testing for authenticity with a challenge-and-response sequence.

Embedded HW/SW License Management

Reference designs, which are subsequently licensed and possibly manufactured by third parties, require barriers to prevent unauthorized use of the intellectual property. For revenue reasons, it is also necessary to track and confirm the number of reference uses. A preprogrammed SHA-256 authenticator (with a secret, user memory, and settings installed prior to delivery to the third-party manufacturer), such as the DeepCover Secure Authenticator (DS28E25), easily solves these requirements and more. As a power-up self-check, the reference (**Figure 6**) performs an authentication sequence with the DS28E25. Only a DS28E25 with valid secret, known only by the licensing company and reference electronics, will be successful at reply with a valid MAC. The reference processor would take appropriate, application-specific action if an invalid MAC is detected. The additional benefit of this approach is the ability to selectively license and enable reference features and functionality through settings stored in the secure DS28E25 memory (for more on this concept, see section Soft-Feature Management).

The DS28E25, or other secure authenticator, with a valid secret is supplied to the licensee or third-party manufacturer through one of two secure methods: 1) preprogrammed by the company licensing the reference, or 2) preprogrammed by Maxim per the licensing company's input and then delivered to the third-party manufacturer. In either case, the number of devices sent to the licensee or manufacturer is known and can be used to validate license fees.

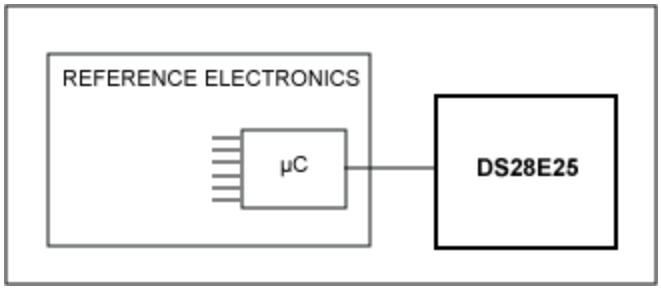

Figure 6. Authenticating the reference design.

Verification of Hardware Authenticity

When verifying hardware authenticity, there are two cases to be considered (**Figure 7**): 1) a cloned circuit board with an exact copy of the µC firmware or FPGA configuration, and 2) a cloned system host. The SHA-1-based DS28E01-100 is used in this example.

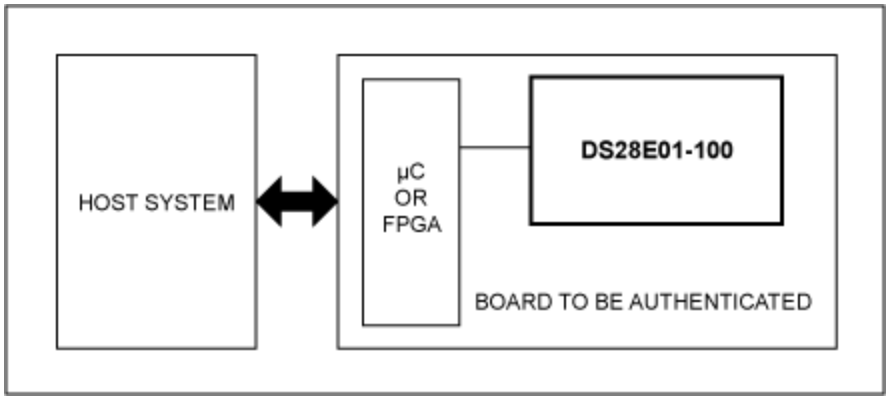

Figure 7. Hardware authentication example.

In the first case, the firmware or FPGA attempts to authenticate the cloned circuit board. For this to be successful, the clone manufacturer must load a secret into a secure authenticator in order to write data into the user EEPROM. While this may make the data look correct, the secret is not valid in the system. Due to the complexity in changing the firmware or FPGA and the need to remain compatible with the host, the firmware or FPGA configuration has to be an exact copy of the original. If the board performs a challenge-and-response authentication of the DS28E01-100 during the power-up phase, the MAC generated by the DS28E01-100 will be different from the MAC computed by the firmware or FPGA. This MAC mismatch is strong evidence that the board is not authentic. This would be detected by the system performing a challenge-and-response sequence with the board, and application-specific action would then be taken.

In the second case, the circuit board authenticates the host system. The board can verify the authenticity of the host using the following procedure: 1) generate a challenge and let the DS28E01-100 compute a challenge-and-response authentication MAC; and 2) send the same MAC computation input data (except for the secret, of course) to the network host, which then computes and returns a challenge-and-response authentication MAC from that data and its own secret. If both MACs match, the board can assume that the host is authentic.

Soft-Feature Management

Electronic systems range from handheld products to units that fill several racks. The larger the unit's size, the more costly it is to develop. To keep the cost under control, there is a desire to construct a large system from a limited selection of smaller subsystems (boards). Often, not all features of a subsystem are needed in the application. Instead of removing these features, it is more cost-effective to leave the board as is and to simply disable them in the control software. This, however, creates a new problem: a smart customer who needs several fully featured systems could just buy one fully featured unit and several units with reduced features. Then, copying the software, the simpler units behave like the fully featured unit but for a lower price, shortchanging the system vendor.

A Maxim SHA-256 device, such as the DeepCover Secure Authenticator (DS28E22), on the board of each subsystem can protect the system vendor from this type of fraud. In addition to serving for challenge-and-response authentication, the same DS28E22 can store the individual configuration settings in its user EEPROM. As explained later in the section Data Security, this data is protected from unauthorized changes, giving full control to the system vendor. The configuration settings can be stored in the form of a bitmap or code words as deemed appropriate by the system designer.

Secure Authentication Device Overview

General Device Architecture

The SHA engine of SHA-1 and SHA-256 devices can be operated in three different ways depending on the operation to be performed. In all cases, the engine receives input data and computes a MAC result. For each operation type, there are differences in the SHA engine's input data based on the targeted use of the MAC result. For any SHA operation, as a fundamental requirement of symmetric key-based secure systems, the host must either know or be able to compute the secret stored in the slave device in order to be authenticated.

Note: given the secure and application-sensitive nature of secure authentication products, device details are omitted from this document. The full version of individual device data sheets, available under a nondisclosure agreement (NDA), provides this information.

Challenge-and-Response Authentication MAC

The primary purpose of SHA-1 and SHA-256 secure authenticators is challenge-and-response authentication. The host sends a random challenge and instructs the slave device to compute a MAC response from the challenge, the secret, user memory, and additional data that together constitute the "message" (**Figure 8**).

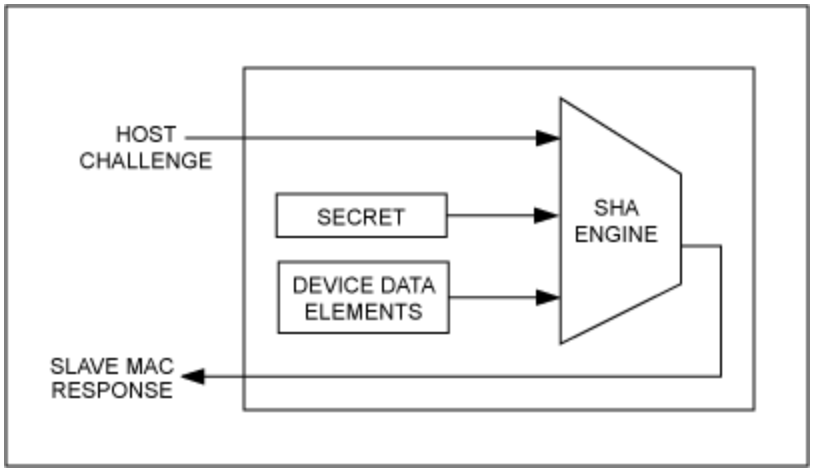

Figure 8. Data flow for challenge-and-response authentication MAC.

After it has finished computing, the slave device sends its MAC to the host for verification. The host then duplicates the MAC computation using a valid secret and the same message data that was used by the slave. A match of the MAC received from the slave provides authentication of the device, since only an authentic slave will respond to the challenge-and-response sequence correctly. It is crucial that the challenge is based on random data. A never-changing challenge opens the door to replay attacks using a valid static MAC that is recorded and replayed instead of a MAC that is instantly computed by an authentic slave.

Data Security

Beyond proving authenticity, it is highly desirable to know that the data stored in the slave device can be trusted. For this reason, write access to the EEPROM in a secure authenticator is securely restricted. Before copying data from an input buffer to the EEPROM or control registers, the slave device requires the requesting host to supply a valid write-access authentication MAC to prove its authenticity. The slave device computes this MAC from the new data in its input buffer memory, its secret, and additional data, as shown in **Figure 9**.

An authentic host knows, or is able to compute, the secret and is able to generate a valid write-access MAC. When receiving the MAC from the host during the copy command, the slave compares it to its own result. Data is transferred from the input buffer to the destination in EEPROM only if both MACs match. Of course, memory pages that are write-protected cannot be modified, even if the MAC is correct.

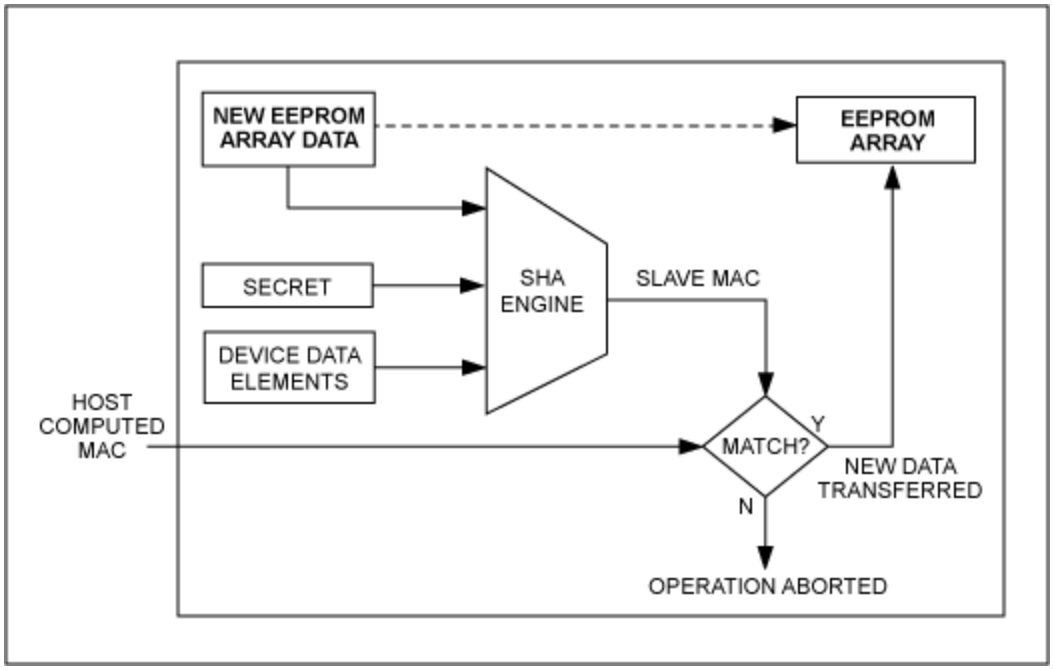

Figure 9. Data flow for a write-access authentication MAC.

Secret Protection

The architecture of Maxim's secure authenticators allows direct load of a secret into the device. Secret protection is provided by both read-protection and, if desired, write-protection, which prevents the secret from ever being changed. This level of protection is effective as long as access to the secret is secure and controlled during initial installation at the equipment production site.

The quality of the secret can be increased in various ways: 1) let the slave device compute its secret; 2) let the slave compute its secret in multiple stages performed at different sites; 3) create device-specific secrets by including the unique device ID number in the computation of the secret; or 4) a combination of 2 and 3.

If each secure authenticator computes its secret, only the ingredients of the secret are known but the secret itself is never exposed. If the secret is computed in multiple stages using different sites, only the local ingredients of the secret are known, which provides a method to control the knowledge of the "final" secret. If the secrets are device-specific, an additional computing step is required for the host, but the potential damage is minimal if a device secret is accidentally discovered. If the secret is computed in multiple stages and made device-specific, the highest possible secrecy is achieved. However, the hosts, like the slaves, need to be set-up at different sites to prevent compromise of system secrecy.

If instructed to compute a secret, the secure authenticator uses its SHA-1 or SHA-2 engine and computes a MAC using device-specific data items as shown in **Figure 10**. The MAC result is then used to derive the new secret.

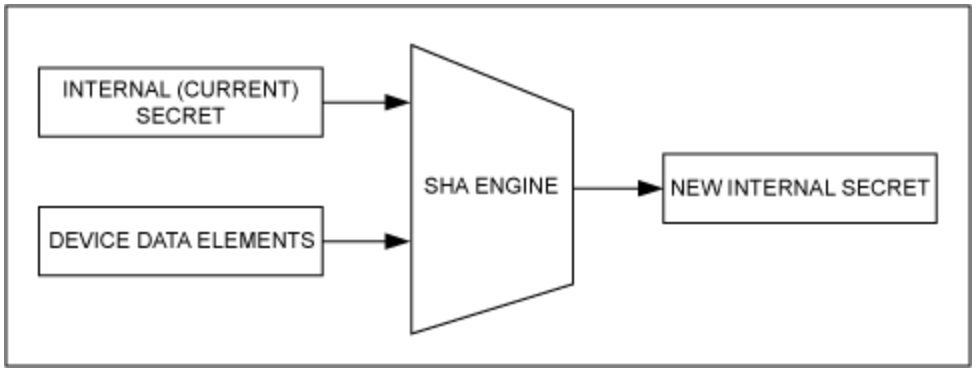

Figure 10. Data flow for new secret computation.

Conclusion

Maxim's secure authentication solutions enable a system developer to protect against inevitable accessory or subsystem counterfeiting attempts. Additionally, tamper-proof user memory provides secure methods to enable or disable features in a system with configurable functionality. A small but powerful silicon chip can make a big difference to the bottom line.

Related Parts		
DS1961S	iButton 1Kb EEPROM with SHA-1 Engine	
DS1963S	iButton Monetary Device with SHA-1 Function	
DS2460	SHA-1 Coprocessor with EEPROM	Free Samples
DS2465	DeepCover Secure Authenticator with SHA-256 Coprocessor and 1-Wire Master Function	Free Samples
DS28E01-100	1Kb Protected 1-Wire EEPROM with SHA-1 Engine	Free Samples
DS28E02	1-Wire SHA-1 Authenticated 1Kb EEPROM with 1.8V Operation	Free Samples
DS28E10	1-Wire SHA-1 Authenticator	Free Samples
DS28E15	DeepCover Secure Authenticator with 1-Wire SHA-256 and 512-Bit User EEPROM	Free Samples
DS28E22	DeepCover Secure Authenticator with 1-Wire SHA-256 and 2Kb User EEPROM	Free Samples
DS28E25	DeepCover Secure Authenticator with 1-Wire SHA-256 and 4Kb User EEPROM	Free Samples

More Information
For Technical Support: http://www.maximintegrated.com/support
For Samples: http://www.maximintegrated.com/samples
Other Questions and Comments: http://www.maximintegrated.com/contact

Application Note 3675: http://www.maximintegrated.com/an3675
TUTORIAL 3675, AN3675, AN 3675, APP3675, Appnote3675, Appnote 3675