Datakey

The power of memory. Secured.

White Paper

Use Cases for Datakey CryptoAuthentication™ Memory Tokens

By: Dean Gereaux
President, Golden Bits Software

Table of Contents

INTRODUCTION

Datakey CryptoAuthentication™ memory tokens are a versatile tool for securely transporting data, unlocking devices, enabling features, storing log data, and more. The built-in Microchip ATECC608B[1] and ATSHA204A CryptoAuthentication ICs enable stored data to be authenticated, protected, and encrypted, providing a high level of security. Data is stored in slots ranging in size from 32 to 416 bytes, depending on which CryptoAuthentication IC is used. An especially important feature is the ability for the host to authenticate the memory token. This feature prevents unauthorized devices from connecting and spoofing a memory token. The memory tokens also have a propriety form factor making it even harder to physically spoof a memory token.

The following use cases illustrate just a few of the uses for CryptoAuthentication memory tokens. Depending on how the memory token is used in a particular application, these use cases, or parts thereof, can be combined. For example, the token may be used to securely track and limit the number of times a device can be used (see Usage Counter section, below), while at the same time the token can securely log other data (see Remote Logging section, below).

SECURE FEATURE DISTRIBUTION

New or enhanced product features can be enabled by sending a CryptoAuthentication memory token to customers. For example, a customer could purchase a feature-update memory token to unlock or turn on additional functionality. The manufacturer can enable these additional features by securely storing the feature information in one or more of the secure slots. These "feature slots" would be protected by a read/write key or an AES key embedded in the memory token's IC; both could be derived from a PIN or password.

The manufacturer would configure the memory token's slots to store an encryption key and/or a read/write key used to protect the encryption key transmission from the memory token's IC to the host. When the feature-update token is inserted into the device, the firmware would securely extract the feature information. The feature information can also be digitally signed by the manufacturer, enabling authentication and proving the feature information has not been tampered with. To authenticate the feature information, the device would either use an ECC signature or HMAC digest. The difference depends on which memory token/CryptoAuthentication IC is used. The ATECC608B uses ECC signatures, whereas the ATSHA204A uses HMAC digests. Either method requires a SHA-256 hash be calculated over the feature information and then verified.

Additional security can be added by creating a unique, customer-specific read/write key, which would prevent multiple customers from sharing the same feature file. This key could be derived from the customer's name, ID, or even a license number.

[1] At the time of this writing, Datakey IAT10.5Kb and IAX10.5Kb CryptoAuthentication memory tokens still used Microchip ATECC608A ICs.

REMOTE LOGGING

Remote embedded systems without network connectivity can use a CryptoAuthentication memory token to securely save log and diagnostic data for later retrieval. Data would be stored in one or more slots; each slot is protected with a read/write key. Any attempt to read a protected slot without the correct key will result in garbled data being returned. This may sound strange, but it is intentional. By not returning an explicit read error a potential hacker has no idea if the data read is valid or not. The read/write key is stored in a dedicated slot and cannot be read or tampered with. When reading an encrypted data slot, the read command uses the slot number of the read/write key, the key itself, and a unique nonce. This information is combined into a hashed value used to decrypt the data during the read operation. A similar process is performed when writing data to an encrypted slot.

When used for remote logging, the memory token is inserted into the host. To ensure the token is valid, the host would authenticate the token using a Challenge-Response method, implemented by the MAC command. When logging, the host can derive the read/write key from different configuration settings on the memory token or use a pre-defined key, possibly tied to the specific remote location. If the host itself contains an ATECC608B or ATSHA204A IC, the authentication can be performed by a shared key that is securely stored in both the memory token and the host.

Memory tokens can be distributed to the remote devices for later retrieval. This use case is also valid when the host system is not remote, but the cost of network connectivity is too high, there is a security concern when enabling network connectivity, or when local storage of data is desired as a backup in case network connectivity is ever lost.

USAGE COUNTER

A CryptoAuthentication memory token can be used as a secure counter for billing purposes or capping the number of times a device or software module is used. A secure memory counter can also be used in a contract manufacturing environment to limit the number of units built, preventing excess units from being illegally produced and sold.

For billing purposes, the counter is incremented for every operation. The counter is then read, remotely or locally, to generate an invoice. The secure counter cannot be tampered with, preventing cheating on billing usage. For capping the number of uses or products built, a counter can be limited to a maximum value after which the counter operation will fail.

For IAT10.5Kb and IAX10.5Kb memory tokens (that use the ATECC608B IC[1]), two counters are available for use, each with a maximum possible value of 2,097,151. Counter 0 can be hard-limited, meaning incrementing beyond the hard limit is impossible and generates an error. The second counter can also be used for counting; however, it cannot be limited. The IAT4.5Kb and IAX4.5Kb memory tokens (that use the ATSHA204A IC), can use the OTP zone, key usage, or writing the count into a protected slot. For key usage, when executing a specific command for slots 0-7, a single byte for counting is incremented, up to 256. This is useful when counting specific items versus a general counter for all items (as the ATECC608B uses). The 512 bits of the ATSHA204A IC's OTP zone can also be used as a counter; each bit transitions from 1 to 0 indicating a count and can never be changed after this transition.

The counter memory tokens would be configured as desired and could optionally be linked to a specific device to prevent cloning. To lock a token to a device, specific device information, such as a serial number would be written into a slot or the OTP area and locked down, making the information immutable.

BIOMETRIC KEY

Embedded systems that utilize access control often utilize a PIN or a password (PIN/Pass) as a way of identifying users. A PIN or password (on its own) is the least secure method of confirming a person's identity, as it is not overly difficult to observe the PIN/Pass being entered. The other weakness with a PIN/Pass is that once stolen, it is easy to share this information with many people. A more secure method of access control is to use a physical credential, such as a mag stripe badge, a proximity card or even a Datakey CryptoAuthentication memory token. Each credential typically has a unique number that is associated with a particular user. Often, this number is stored electronically and may even be encrypted, making it much more difficult for someone to obtain this information. Additionally, since each credential is unique, a lost or stolen credential can only be used by one other person (provided it cannot be cloned). Biometric data, such as an eye or fingerprint scan, provide an even higher level of security in accurately identifying an individual and is nearly impossible to circumvent. Even more secure than any one of these authentication methods is to use two or more of them. This is referred to as dual-factor or multi-factor authentication. For example, a system might require both biometric data plus a PIN/Pass for applications where it is absolutely necessary to correctly identify a person with a high level of certainty.

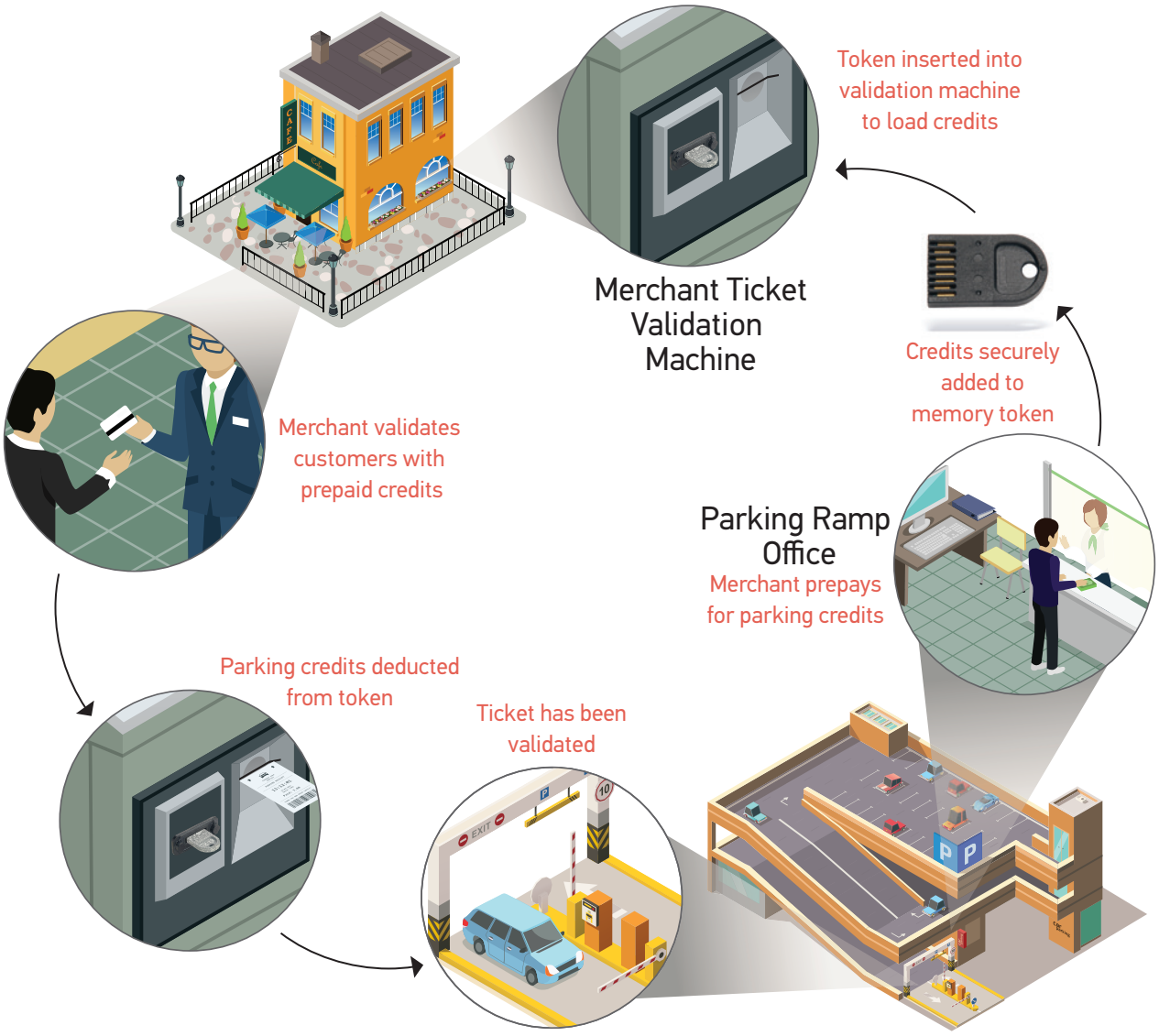
Another form of multi-factor authentication would be to use biometric data along with a physical credential, like a Datakey CryptoAuthentication memory token. In this scenario, the memory token provides an added benefit in that the memory token can store a person's biometric data which can only be unlocked with the correct PIN/Pass. This utilizes all three access control methods, a PIN/Pass, a physical credential, and biometric data. This implementation also has the advantage of not requiring a network connection, as the credential information (PIN/Pass, biometric data) is stored on the memory token rather than on a remote server. Remote sites with no network connectivity or other scenarios with spotty or unreliable network connectivity benefit from securely storing this credential information right on the user's credential. This biometric key could be issued to contractors, employees, or any individual whose identity must be confirmed with near certainty.

Operationally, the memory token is inserted into a secure reader and is used in conjunction with a biometric reader (for example, a fingerprint reader or an eye scanner). The biometric data from the scanner is compared to the biometric data stored on the memory token. The data on the token may be encrypted or protected with a read/write key. Additionally, the system may also prompt the user to enter a PIN/Pass. When the memory token was initially configured, an encryption key and hash of the PIN/Pass would be stored in two separate slots within the token's CryptoAuthentication IC. To verify the PIN/Pass, a hash of the PIN/Pass is combined with several other information fields on the IC. This is then used to generate a second hash along with a random challenge. This newly created hash is used with the CheckMac command to verify the PIN/Pass. The PIN/Pass hash can also be used to generate the read/write key used to protect the encryption key. A correct PIN/Pass effectively unlocks the encryption key from the memory token's IC.

The biometric data would also be verified by using an HMAC (ATSHA204A) or ECC signature (ATECC608B). The memory token with an ATSHA204A would derive the HMAC key using a random nonce and the PIN/Pass value. A hash of the biometric data is used as input to the HMAC function. The resulting HMAC digest is then compared to what is expected. If different, the biometric data has been tampered. For memory tokens with an ATECC608B, an ECC keypair is generated and is used to sign the biometric data. The signature is then securely stored in a slot. To verify the data, a hash is generated and verified using the previously stored signature and public key of the ECC keypair. The ATECC608B securely performs all of the necessary ECC operations.

SECURELY TRANSFERRING CREDITS

Credits or usage allotment for a device or application can be securely transferred using a Datakey CryptoAuthentication memory token. A parking garage metering system could use the token to sell/transfer parking passes or credits to merchants using the garage, and the merchants, in turn, would validate parking for customers or employees. The merchant would have a device to validate parking tickets. The memory token is inserted into this device, thereby loading the device with the number of validations purchased. As parking tickets are validated, the credits on the memory token are decremented. Unused credits could potentially be refunded to the merchant. The memory token could be optionally linked to a specific merchant and the credits could even have an expiration date. The following diagram illustrates the above process:



How are parking credits securely added and used? The number and type of credits would be stored in one or more secure slots. For example, $500 of parking credits would be stored in Slot 0. The slot would be protected by a read/write key stored in Slot 1. The read/write key would be derived from the Merchant's PIN code used to access the memory token. To add credits, the new dollar amount is added to the Slot 0 value. To use credits the dollar amount is subtracted from the current Slot 0 value. To handle unexpected interruptions form power loss or another event, slot 2 can be used to record the transaction before committing the changes to Slot 0, a simple transaction journal.

SECURE VENDING INVENTORY CONTROL

In a large manufacturing environment, high-value parts, tools and supplies are often secured and tracked. These could be things like personal protective equipment (PPE), batteries, welding rods or other shop supplies. To minimize loss and track usage of these items, manufacturers will sometimes use industrial vending machines to control access to these items. In order to get an item from an industrial vending machine, cash is not used. Instead, the employee must identify themselves to the machine. Some systems may simply require a PIN or password to be entered, but this (on its own) is not very secure, as a PIN/password is easily witnessed and can be shared with many people. Having a secure credential, in the form of a CryptoAuthentication memory token, is much more secure and may be more reliable and practical than using a biometric sensor in a potentially harsh manufacturing environment. In addition to simply granting access and sharing the identity of the employee, the memory token could also be configured to restrict the type of parts or the number of supplies vended for each user. The vending machine could also be configured to block lost or stolen memory tokens.

In one possible implementation, each slot in the memory token would contain the items allowed, vend count, and a last vend timestamp. The slots would be protected with a read/write key which would be uniquely generated for each token based on the token's serial number and configuration information. Each token could also contain a unique authentication key securely locked in a slot, which would be used to authenticate the token itself. The user is verified by storing a hash of the user's PIN or password into a slot which cannot be read or tampered. To verify a user/a user's token, the user enters their PIN/Password, which is combined with other token information to generate a unique random hash used to verify the stored PIN/Password. The CheckMac command verifies the PIN/Password by returning a Yes/No response.

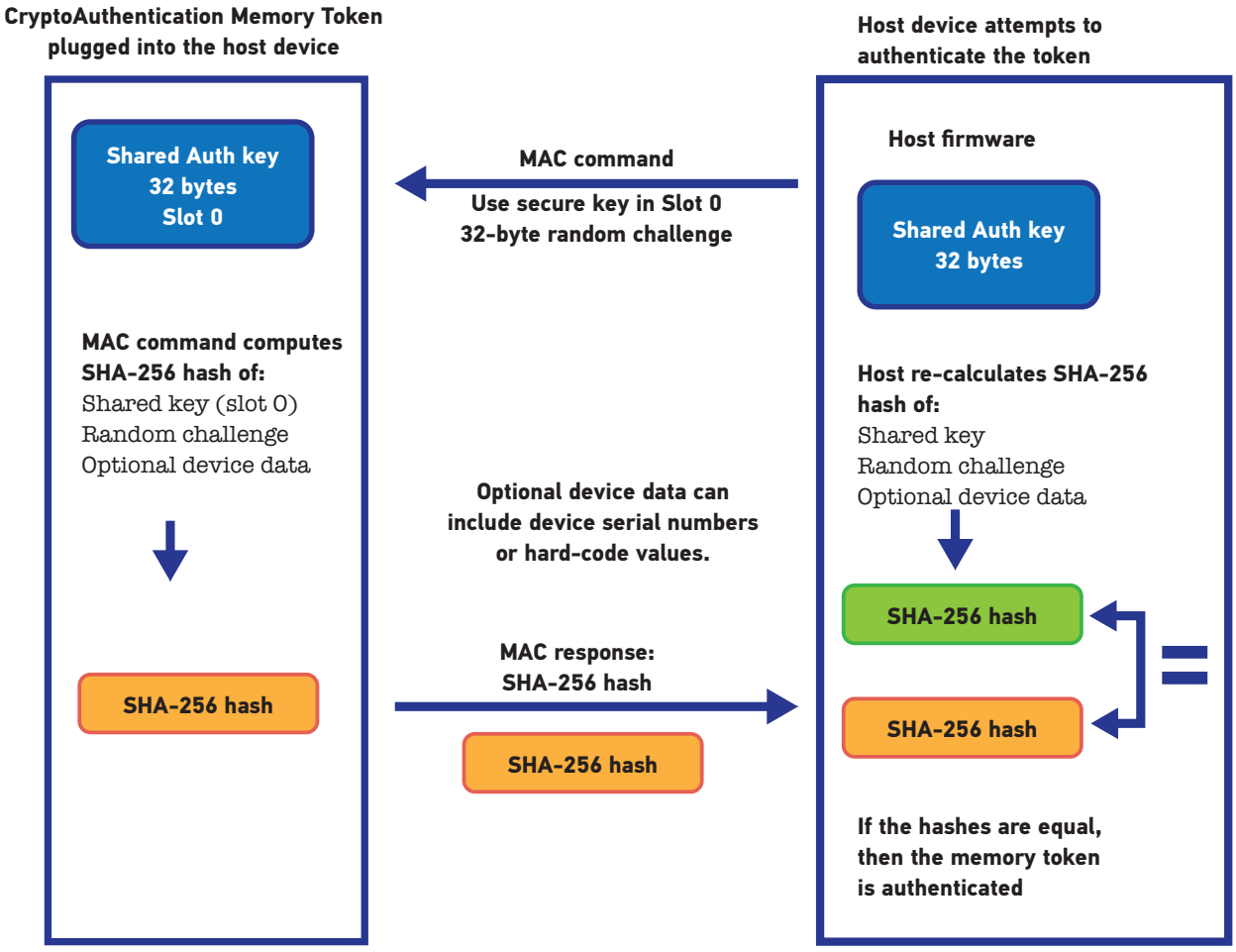
To initially create the token for a user, the vending machine (or another dedicated device) would set the user's PIN/Password, the allowed items, the authentication key, and possibly a token expiration date into the memory token slots. When the token is created, the OTP area can be used to store information that does not require encryption; items such as token creation date/time, expiration date, location, and username. The OTP area is not encrypted, however once locked it cannot be changed.

When inserted into a vending machine, the machine authenticates the memory token. This proves the memory token is in fact valid. Secondary checks could include checking the expiration date and username. After these initial checks, the user enters their PIN/Password into the vending machine's keypad. This is checked using the PIN/Password stored on the memory token. After all these authentication steps, the vending machine would then process the user's request and update the memory token as necessary.

MEMORY TOKEN AUTHENTICATION

While technically not a use case or application, it is important to highlight the ability of the host to authenticate the memory token. The "host" is the embedded device the memory token is inserted into. The ATECC608B[1] and ATSHA204A ICs used in Datakey CryptoAuthentication tokens authenticate by using a 32-byte key shared between the memory token and host. This secret key, along with a random challenge from the host, plus optional data, is hashed with secure IC device-specific information. The result (another hash) is then sent back to the host where it is compared to the expected hash value. The optional data enables additional diversification when generating the hash.
The important concept here is the shared auth key is securely stored on the memory token. It cannot be read or tampered. The memory token proves it has possession of the shared auth key by correctly generating a hash given a random host challenge. A random challenge is critical in authenticating, otherwise a rouge device can simply replay a previously captured sequence. A random challenge forces the device to generate the correct hash versus returning a canned response.

The following illustrates the authentication steps.



The 32-byte random value is sent by the host to the memory token, which computes a hash of the random value and the token's Auth key, which should match the Auth key that is securely stored in the host. In order to securely store the Auth key, the host can use an on-board CryptoAuthentication IC, another CryptoAuthentication memory token or some other secure method such as Arm TrustZone or another dedicated secure storage device. The MAC command is executed by the memory token using the securely stored shared key to calculate the correct hash response. The host then compares this hash to the hash it calculated. If they match, then the memory token is authenticated.

CONCLUSION

This paper highlights just a few of the potential applications for Datakey CryptoAuthentication memory tokens. The functionality described in each use case can be combined or removed to fit a particular application. Do you need to store sensitive data? If so, then using a protected slot is recommended. Do you need to authenticate a device? Then using a shared key with the MAC command is recommended. In summary, start by asking what features and functions are needed. Are there any specific requirements? From this top level, then drill down by asking more questions and developing use cases specific to your application.

ABOUT THE AUTHOR

Golden Bits Software® is a software consulting firm based in San Diego, CA, whose principal is Dean Gereaux. Experienced in a wide range of technologies and platforms, including security and Microchip's CryptoAuthentication devices, past projects include securing wireless building lighting controls, medical devices, and full PKI creation and integration with IoT devices.

Golden Bits® is a Microchip Design Partner for security devices.

Contact information:
Email: deang@goldenbits.com
Phone: +1-858-259-3870

Web: goldenbits.com

Golden Bits® SOFTWARE

GLOSSARY

TERM	DEFINITION
Challenge-Response	A simple method for authenticating a peer using a shared key. Both peers send a random challenge (i.e. Nonce), and each peer hashes the challenge with the key. If both peers generate the expected hash, then they have authenticated each other.
CheckMAC command	ATECC608B and ATSHA204A command used to validate a hash generated using a shared key and random challenge.
Digest	A digest or message digest is the result of a message being passed through a one-way cryptographic hashing formula. Message digests protect the integrity of a message/piece of data, as if any part of the message/data is changed the digest will also change.
ECC Key	Elliptical Curve Cryptography asymmetric key used to sign and verify data. Asymmetric keys are composed of a private and public key pair linked by the underlying elliptical curve math. The ATECC608B supports ECC secp256r1 curve. The ATSHA204A does not support ECC keys.
ECC Signature	A signature generated by an ECC private key using the hash of the data to sign. For the ATECC608B, the signature is 64 bytes in length. The corresponding public key is used to verify the signature.
Hash	A cryptography function than computes a fixed size unique result from an arbitrary amount of input bytes. For example, a SHA-256 hash of 10 Mbytes would produce a value of 32 bytes, essentially a unique thumbprint of the 10 Mbytes. If one bit of the 10 Mbytes is changed, the hash value will also change.
HMAC	Hashed Message Authentication Code. A type of hash combined with a secret key to generate a secure hash.
MAC command	ATECC608B and ATSHA204A command used to create a hash response using a secret shared key and a random challenge.
Nonce	A random value intended for one time use, usually a byte array of 32 bytes.
OTP	One Time Programming. An area of the ATECC608B and ATSHA204A which is set once and locked. Once locked the area cannot be changed.
SHA256	A type of hash. The ATECC608B and ATSHA204A both support SHA256.

204-0010-000 Rev. A 9/21

ATEK Access Technologies
10025 Valley View Road, Ste. 190
Eden Prairie, MN 55344 U.S.A.
PH: 1.800.523.6996
FAX: 1.800.589.3705
+1.218.829.9797
www.atekaccess.com