Application Note 154
Passwords in SHA Authentication

www.maxim-ic.com

This Application Note describes techniques that can be used in systems using Dallas SHA iButtons® as network user authentication tokens. These techniques are used to bind user passwords into the SHA-based authentication scheme. The reader should be familiar with the standard Dallas SHA iButton authentication devices and methods. See Application Note 149, SHA 1-Wire® Device Application Guide, for more information and general applications information.

INTRODUCTION

The Dallas 1-Wire SHA devices provide a highly secure way to be sure that a user iButton token is authentic. However, some applications demand a higher level of security in which the person who brings the token (the user) is also authenticated. This is known as "something you bring, something you know" security. The "something you bring" part is the SHA-enabled iButton token. The "something you know" part is a PIN, password, or pass phrase that is committed to the user's memory.

This Application Note describes a method whereby a user-provided password can be integrally bound to the secure authentication process. This makes the authentication of the iButton token dependent on information provided by the user (as opposed to the service provider) yet does not require that any user passwords be stored in a database at the server, or that they be known to the service provider at all.

In the remainder of this document, we will refer to a "user password". In fact, this could be a PIN, password, or a more complex pass phrase (or even a digest of a pass phrase). We will assume that this information is collected from the user through the keyboard of the agent computer (workstation or terminal), although it could come into the system by other means.

BINDING THE USER PASSWORD

To make the user password an integral part of the iButton token authentication process, it must be bound-up with the authentication secret stored in the SHA iButton token. In the high-security iButton authentication schemes already defined, a master authentication secret is hashed together with the address (serial number) of the token to create a working secret which remains hidden in the token. This secret is unique to the token because it is entirely dependant on the unique address of the token.
(See *Application Note 152, SHA iButton Secrets and Challenges,* http://dbserv.maxim-ic.com/appnotes.cfm?appnote_number=835.)

To bind the user password to the token in a similar manner, the user password and the unbound token secret are hashed to create a MAC, which in turn becomes the new secret. This new secret is written over the old one, and the iButton token is now forever bound to both the token address and the user password. Both the token and the user password are now required to perform authentication. Because the original secret is obliterated in the process, this new password-bound secret can never be removed or changed, except by the service provider.

iButton and 1-Wire are registered trademarks of Dallas Semiconductor.

AUTHENTICATION WITH BOUND PASSWORDS

The actual authentication function when secrets are bound to passwords is little more complex than with unbound secrets. The server must know the iButton token serial number and the user password to compute the bound secret for the token, which in turn is required to authenticate the token. When the user presents the token for authentication, the agent (the user's terminal or computer) must ask that he provide the password. The password must then be transmitted, along with the other token information, to the server so that it can be used in the authentication process.

However, sending the password over an insecure link to the server would expose the system to eavesdropping attacks. So, the password must be encrypted before being sent to the server. One way to encrypt a small message is to use the SHA engine in the token itself to generate a one-time pad against which the message (the password) can be encrypted using a simple exclusive-OR logical function. (See *Application Note 150, Small Message Encryption using the DS1963S iButton*.) Of course, any reliable symmetric encryption algorithm could be used to protect the password, or a secure network connection could be used, like SSL (Secure Sockets Layer).

The authentication process, using the SHA device to encrypt the password, is as follows:

1) The user arrives at a workstation, presents his iButton token and requests access.
2) The token data is read and transferred to the server by the workstation.
3) The server generates a random challenge and sends it to the workstation.
4) The workstation requests the password from the user.
5) The workstation uses the user iButton token, the random challenge and the token address to generate a MAC.
6) The workstation Exclusive-ORs the MAC against the password and sends the resulting encrypted message to the server.
7) The server generates the same MAC, given the token address and the challenge value, and thereby decrypts the message to obtain the original password.
8) The server uses the token address and the master authentication secret to regenerate the user token's original unique token secret.
9) The server uses the original unique token secret and the user password to regenerate the bound secret, and then uses that to complete the normal authentication of the user and his iButton token.
10) The server discards the regenerated user password and secrets.

This method has the following features:

- All network messages are secure and, because of the random challenge component, are entirely different every time an authentication is performed. There is no value in eavesdropping.
- Authentication of the user iButton token and his password is secure. No alteration of the network data can cause a false authentication. Any alteration is detected and causes authentication to fail.
- The user password and iButton token are bound together by an alteration of the secret in the token. The central server does not need to maintain a database of user passwords, nor does it have to perform a user password lookup when authenticating a user.
- The user password cannot be extracted from the iButton token. It does not exist in the iButton at all.
- The terminal or workstation does not have to perform encryption, decryption, or have secure networking capabilities for the scheme to nonetheless be secure against attack.

TOKEN AND PASSWORD DEPLOYMENT

The user password must be installed in the user iButton token before it can be used. This can be done when the iButton token is prepared for the user. However, password binding does not necessarily have to be done by, or in the presence of, the service provider. The user can be given the iButton token in an unbound state and use his home computer or a workstation to embed his personal password, or elect not to embed a password at all (if the system permits this option).

The user password is bound-up in the token secret, so it cannot be changed without access to the system master secret, or at least to the unique token secret. In most situations, the easiest method for changing a user password is for the service provider to simply generate a new token for the customer, transfer the data contents from the old token to the new token (which may involve re-signing the data), and then obliterate the data and secrets in the old token and recycle it. The customer then injects the desired new password into his new iButton token. (See *Application Note 152, SHA iButton Secrets and Challenges*, http://dbserv.maxim-ic.com/appnotes.cfm?appnote_number=835.) This method preserves the security of the secret sharing schemes that the service provider may have in place because the system master secret is not required, but only ready and prepared new iButton token.

Because the user iButton token does not contain the sensitive master system secret, it could be prepared for use remotely through the network. The server would have to authenticate the user and token by other means, of course (sometimes called "twenty-questions authentication"), and then use the token serial number to generate and un-bound unique token secret which it would send to the user token using a secure connection. Next, the user would provide a password and the resulting secret would then be bound to the password locally.

SECURITY OF PASSWORDS IN HUMAN HANDS

User passwords are generally considered low security when compared to cryptosystems that authenticate electronic tokens. Humans are notoriously insecure containers for secrets, and user interfaces are rarely able to protect the sensitive information from exposure. There are a variety of attacks from simple "shoulder surfing" all the way to illicit alteration of the agent software so that it will capture passwords and deliver them to the attacker. (Not to mention personal attacks such as confidence scams, drugs, bribes, and coercion.) Because user passwords must always come into the public domain at some point, (i.e., when entered into the workstation), they are never generally very secure. Nonetheless, user PINs or passwords are the only workable way that we have to authenticate people.

The methods described herein also must assume a few things about the user's agent (the workstation):

1) That the user's agent (workstation, PC, or terminal) will accept his password, encrypt it, and send it to the server with reasonable security.
2) That the user's agent will not record the user password or write it to a media where it could later be compromised.
3) That the users agent will not relay his password to other parties or use it for any other purpose.

In other words, the user must trust the agent to which he divulges his password to be secure and to protect the password he has provided. It would be a good idea to arrange for agent software to be signed and tested for authenticity when used to preclude many of these attacks. Authentication of the agent by the server prior to authentication of the user token by the server would be prudent.

PASSWORD PRE-PROCESSING

Because the cryptosystem requires that the password be an exact match, passwords provided by users must be subjected to pre-processing before they are used in the authentication process. Users may enter leading spaces, trailing spaces, incorrect cases, or other mis-formatted characters. These errors generally do not detract from the validity of the password, but would cause an authentication failure, and user frustration, if used as entered.

If mixed case passwords are allowed, then users may have trouble entering the correct case despite keying the correct characters. Converting all passwords to upper case reduces the size of the password field but makes human password entry less sensitive to human error. Leading and trailing spaces should generally be stripped from passwords, and obviously illegal characters may be detected early to allow the user an opportunity to re-enter the password before the entire authentication process is launched. Password length limitations (minimum and maximum) must be enforced, and some systems may wish to impose rules that require passwords to contain letters and numbers to improve security. Padding passwords to make them fit the required field size is also an important consideration. Padding may consist of nulls, spaces, fixed codes, or even numeric sequences designed to make attacks more difficult. The selection of a padding method may greatly affect the security of the system.

Whatever rules are employed in the pre-processing of passwords, it is important that these rules be applied exactly the same when the original user password is installed as they are when the password is requested for authentication.

CONCLUSIONS

In the vast majority of user authentication systems, user passwords are the sole means of security. Passwords are low cost and simple, but it is impossible to know when a password has been compromised. Keys (mechanical tokens) have an advantage that the user knows when the key is lost. Advanced electronic token systems can deliver greatly improved authentication of the user token, but fail to authenticate the person. Binding user passwords into an electronic token authentication system can enhance the security and gain the benefits of both token-based and password-based security schemes.