

Easy Authenticator App



User Manual

Contents

Table of Figures	4
1. Easy Authenticator App User Manual	5
1.1 What is an Authenticator?	5
1.2 Which Web Servers are Compatible?	5
1.3 Installation	5
1.4 App Settings and User Preferences	6
1.5 Vault Files	7
1.6 Access Controls	7
1.6.1 Password Protection	8
1.6.2 Key File	9
1.6.3 USB Security Key	9
1.6.4 Yubikey Challenge-Response	9
1.7 Vault File Management	10
1.7.1 New, Open, Save, Close	10
1.7.2 Save As	10
1.7.3 Import and Export Accounts	10
1.7.4 Export to Other Formats	10
1.8 User Profiles	11
2. Account Management	12
2.1 Easy Authenticator Tokens	12
2.2 Server Interactions	12
2.3 Register an Account	12
2.3.1 Using Auto-Generated Digital Signature Keys	14
2.3.2 Using a Certificate	16
2.3.3 Using a Signed Certificate	16
2.4 Sign into an Account	17
2.5 Editing an Account on the Web Server (Except the Public Key)	19
2.6 Renewing the Account's Public Key	20
2.7 Digitally Sign a Privileged Operation	22
2.8 Resetting a Public Key	23
2.9 Terminating an Account	23
2.10 Account Deletion	24
3. Easy Authenticator Tools	25

3.1 Yubikey Integration	25
3.1.1 Static Password with the Yubikey	25
3.1.2 Challenge-Response with the Yubikey	26
3.1.3 Generating a Self-Signed Certificate on the Yubikey	26
3.1.4 Importing a Certificate on the Yubikey	28
3.2 USB Security Key	28
3.2.1 What is a USB Security Key?	28
3.2.2 How to Generate a Key	29
3.2.3 How to Use a USB Security Key to Secure a Vault file	30
3.3 Certificate Viewer	30

Table of Figures

Figure 1: Authenticator App Splash Screen.....	5
Figure 2: Easy Authenticator App Settings	6
Figure 3: Vault File Icon.....	7
Figure 4: Security Parameters (Access Controls) for Vault Files	8
Figure 5: Password Management Interface	8
Figure 6: User Profiles Interface.....	11
Figure 7: Example of User Registration Interface.....	13
Figure 8: Authenticator Token for Registration	13
Figure 9: Blank Registration Form.....	14
Figure 10: Filled Registration Form	15
Figure 11: Registered Account in the Vault.....	15
Figure 12: Display of Account Details.....	16
Figure 13: Registration Requiring a Signed Certificate	17
Figure 14: Sign-In Interface.....	18
Figure 15: Sign-In Authenticator Token	18
Figure 16: Sign-In Authenticator Interface.....	18
Figure 17: User Profile Interface.....	19
Figure 18: Account Modification Interface.....	20
Figure 19: Account Modification Authenticator Token	20
Figure 20: Example of a Key Renewal Interface.....	21
Figure 21: Public Key Renew Authenticator Token.....	21
Figure 22: Public Key Renew.....	21
Figure 23: Digitally Signing a Privileged Operation	22
Figure 24: Digital Signature Authenticator Token	22
Figure 25: Digitally Signing a Generic Operation.....	23
Figure 26: Account Termination Interface.....	24
Figure 27: Authenticator Token for Account Termination.....	24
Figure 28: Easy Authenticator App Account Termination Interface	24
Figure 29: Yubikey Settings Interface.....	25
Figure 30: The Yubikey PIV Editor Interface	27
Figure 31: Generating a self-signed Certificate on a Yubikey.....	28
Figure 32: USB Security Key Editor Interface	29
Figure 33: Selected USB Security Key for Vault Access.....	30
Figure 34: Details of a Loaded Certificate.....	31

1. Easy Authenticator App User Manual

1.1 What is an Authenticator?

An authenticator app either performs the tasks associated to authentication on behalf of the user or assists with the tasks to speed up the process and facilitate the interactions. The Easy Authenticator app is based on the use of digital signatures to authenticate the user through a challenge-response protocol, which removes the requirement to generate and share a password with the web server.

To validate a digital signature, the web server needs to have the user's public key. This is what will be provided to the web server at registration, instead of a sensitive password. The public key is not sensitive and does not need to be protected. The user's private key is sensitive and will not be shared with anybody. It will remain under the control of the user at all times.

In the IT world, digital signatures are commonly used to provide authenticity when interacting with entities through a public key infrastructure (PKI) such as the Internet. Organizations use them to protect the integrity of documents. Digital signatures also protect the integrity of cryptocurrency transactions, such as the ones involving Bitcoin.

1.2 Which Web Servers are Compatible?

To use the Easy Authenticator app with a web server, this server must implement the communication protocol required for the interactions. A web server that authenticates users through passwords is not compatible with the Easy Authenticator app. Easy Authenticator can be combined with a second factor of authentication to increase the robustness of the authentication process.

1.3 Installation

The Easy Authenticator is distributed as an App for Windows 10. At installation, you will see a splash screen like the one displayed in Figure 1.

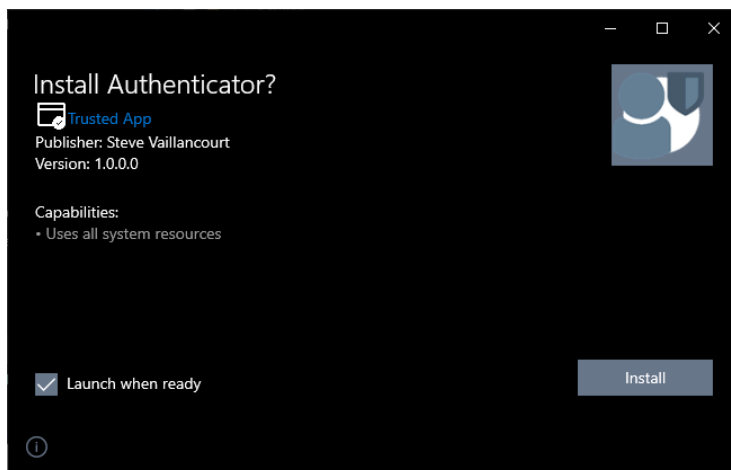


Figure 1: Authenticator App Splash Screen

After installation, the Easy Authenticator app will be integrated into the Windows Operating System and be ready to use.

1.4 App Settings and User Preferences

After installing the Easy Authenticator, it is recommended to edit the user preferences to get the best possible user experience. The settings are accessed from the menu *Edit / Settings...* See Figure 2 for an example of the interface.

The following settings can be edited:

- ***Keep Application on Top***

Easy Authenticator makes use drag-and-drop operations to exchange information back and forth with web servers. To keep the Easy Authenticator app from going in the background during those operations, check the box and it will remain on top of other windows. This menu specifies the user's general preference, but the status can also be quickly edited from the *View / Always on Top* menu.

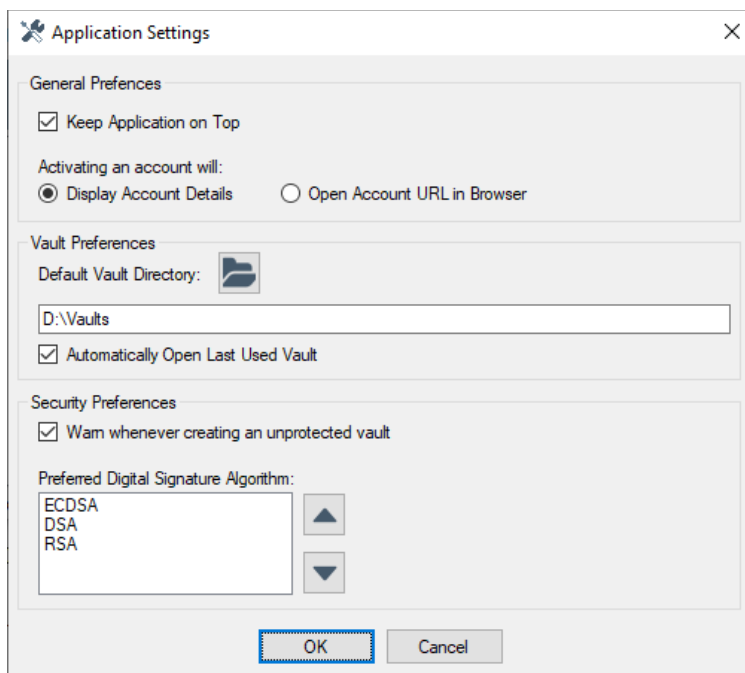


Figure 2: Easy Authenticator App Settings

- ***Activating an account will: Display Account Details / Open account URL in Browser***

Activating an account means triggering it by double-clicking, selecting and pressing the space bar, or any other activation method. This option allows to specify the behaviour of the app when this happens: displaying the account details, or opening the account's URL in the user's default browser.

- *Default Vault Directory*

This value should be set to point to the folder where the Easy Authenticator app should go to by default to save and open vault files. This will define the initial directory, but it is always possible to navigate to whichever directory on your system the file is located or where it should be saved.

- *Warn whenever creating an unprotected vault*

Unprotected vaults have no [security controls](#) applied to them. Even though they will still be encrypted, the encryption is trivial and should not be seen as a real protection. Other actions that can leave the vault file unprotected are when they are [exported to other formats](#). Use this option to request a warning message whenever a vault file is about to be saved without proper security controls applied.

- *Preferred Digital Signature Algorithm*

This option allows to order the digital signature algorithms supported by Easy Authenticator in order of preference. Not every web server will support all three, and when registering a key with them, the key must conform to a supported implementation. The Easy Authenticator app will allow the user to select supported algorithms only. However, if multiple algorithms are supported, the Easy Authenticator app will sort the supported implementations using the order defined by this option. In case of doubt, leaving this setting at its default value is recommended.

1.5 Vault Files

Vault files are used by the Easy Authenticator app to store the account information, including private key information used for digital signatures. This information is sensitive and must be protected. A vault file has the extension **.vault* and is associated to the app at installation. Users can create as many vault files as needed to support their requirements. The icon associated to the file will appear as illustrated in Figure 3.



Figure 3: Vault File Icon

1.6 Access Controls

To protect the account stored in vault files, access controls will be used to generate a symmetric encryption key. If no access controls are used, the encryption key is trivial and offers no real protection. Several access controls can be compounded to create a multi-factor encryption key. See Figure 4 for the vault access interface.

There are two ways to apply access controls to a vault. The first is at its creation, using the *File / New* menu. The other is when an existing vault is saved using the *File / Save as...* menu, which allows the user to save the vault file at a different location and/or with different access controls. When opening a file, the user must provide the same access controls to recreate the symmetric encryption key.



Warning: If the user cannot recreate the access controls to decrypt a vault, it might be impossible to access the account information that it contains: accounts will need to be [reset](#).

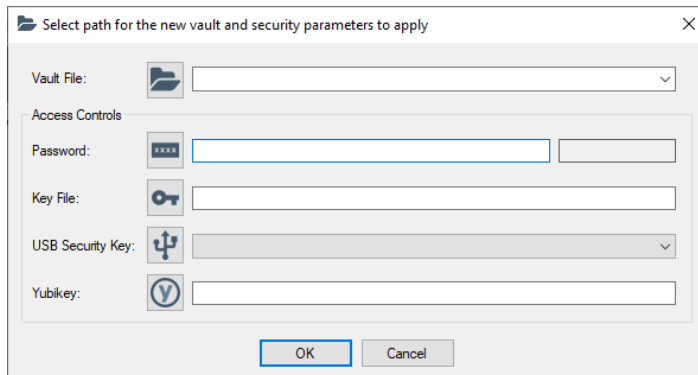


Figure 4: Security Parameters (Access Controls) for Vault Files

1.6.1 Password Protection

The encryption key can be derived from a password applied as a security control. The strength of the key will be correlated with the unpredictability and entropy of the password. To provide insight into the entropy and predictability of the key, the user can press the button next to the password entry box to display a password management interface. An example of the interface is provided in Figure 5.

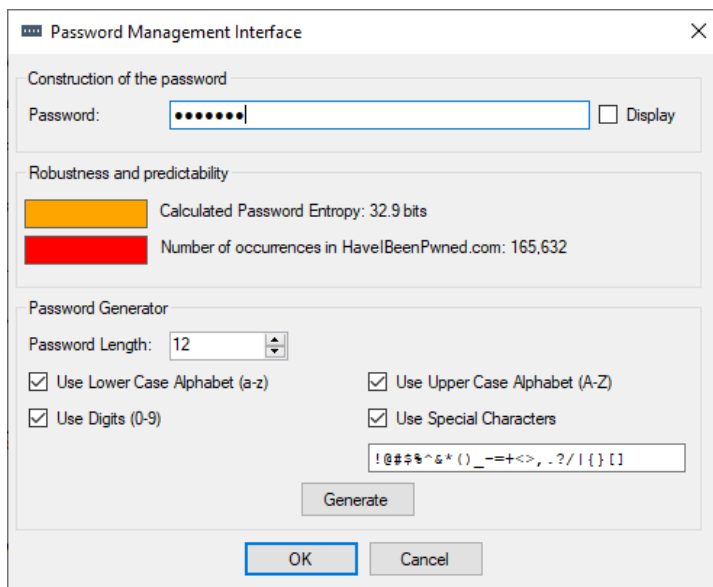


Figure 5: Password Management Interface

The interface uses the public API of the website [HaveIBeenPwned.com](https://haveibeenpwned.com) which researches and catalogues breaches to online accounts. When providing a password, the number of times it appears in known breaches will be displayed.

When protecting a vault file using a password, it is important that it is both unpredictable and presents a high value of entropy (both feedback colors will be green) so that someone that has access to the encrypted vault file will not succeed in decrypting the information.

The interface can also be used as a password generator. Make sure that you can keep track of passwords that you apply to encrypt vault files. A recommended approach is to use a password manager or a hardware device to protect your password. One such device is the Yubikey, which can be configured from the menu *Edit / Yubikey Settings...* for a static password use.

1.6.2 Key File

Another way to create the encryption key is to use a key file. This can be any file of any type provided that:

- The contents never change
- The user has easy access to it to decrypt the vault file
- The file is kept secure and out of reach from unauthorized individuals

The encryption key will be derived from the selected file using an approved hashing function: Secure Hash Algorithm 256 bits (SHA-256). It is advised to keep an offline copy of the key file. Also, it is advisable to select a file that is not “obviously” a key file for an Easy Authenticator vault, unless that file is securely stored and out of reach of unauthorized individuals.

1.6.3 USB Security Key

For your convenience, you can convert a USB key into a hardware-based [key file](#). To make it recognizable as such for the Easy Authenticator app, a USB Security Key must contain two specific files at the root. One contains the identification string of the key and the other contains the key file. Easy Authenticator can assist you in the creation of a USB Security Key (see [3.2.2 How to generate a key?](#)). This key must be handled with caution.

If a USB key is connected to your system and holds the two files to identify it as such, the user can click the icon next to the USB Security Key input to automatically select the first detected USB Security Key. If multiple keys are connected, the user can also use the drop-down list to select the required key.

1.6.4 Yubikey Challenge-Response

Easy Authenticator incorporates the use of a Yubikey. The YubiKey is a hardware authentication device manufactured by Yubico to protect access to computers, networks, and online services. One method of providing access control security to a vault file is to use the challenge-response feature of the Yubikey. The key must be configured for such use beforehand (see [3.1 Yubikey integration](#)). If your key is configured for challenge-response, clicking the icon will trigger the challenge and write the response in the corresponding entry. This can be used to create the encryption key of a vault file quickly and easily. If key requires touch activation, it will blink, and the user will need to press a finger to the key before the response to the challenge is provided.

1.7 Vault File Management

There are several operations that can be accomplished with vault files. Although encrypted, they remain files that can be copied, moved, and renamed from the operating system. Accessing the contents means that the Easy Authenticator app will have to be provided with the security parameters to decrypt the file.

1.7.1 New, Open, Save, Close

To get started with a vault file, the typical approach is to select New, either from the menu or toolbar, and create the vault file with the chosen security parameters. Saving the file updates the content and reapplies the encryption. The user will not need to submit the security parameters every time the vault is save to the file; the app will remember them from when you the file was opened. Closing the file means that the app remains opened, but the vault file will be encrypted and closed.

1.7.2 Save As

The *File / Save As...* menu is used when the user wishes to modify the security parameters and/or the location of the file. The user can overwrite the existing file with this operation, meaning that only the security controls would change. When writing the file to a new location, the original file, with its security parameters, still exists. If it is no longer required, it should be deleted after verifying that the new vault file works as intended.

1.7.3 Import and Export Accounts

It is possible to merge or reorganize vaults using the Export and Import functionalities. To export accounts, they must first be selected in the interface. The menu *File / Export / Selected Accounts...* will allow the user to save a new vault file containing the selected accounts. Specific security parameters for that vault can be applied during this operation.

To add existing accounts from another vault file to the currently opened vault file, the *File / Import / Accounts from vault...* menu is used. All the accounts from that other vault file that do not exist already in the current vault will be imported. Of course, the security controls of the other vault file will need to be provided during the operation.

1.7.4 Export to Other Formats

For convenience, it is possible to export accounts into alternative formats. Using the menu *File / Export / Vault to CSV...* will allow the creation of a **.csv* (*Comma Separated Values*) file containing the details of the accounts. Also, the current vault can be exported to a plain text, readable format using *File / Export / Vault to Plain Text...* These operations are one-way only: files in that format cannot be imported back into a vault.

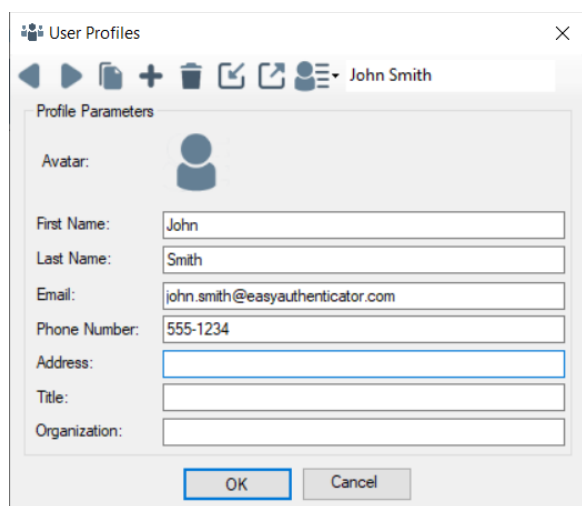


Warning: Vaults exported to CSV or Plain Text are not protected. The information contained in them is directly readable and can lead to disclosure of sensitive data.

1.8 User Profiles

User profiles are part of the settings of the Easy Authenticator app. They allow a user to predefine digital identities for quick registration with web servers. A web server will request several points of information such as email, first and last names, phone number, and date of birth. Some will be required while others will be optional. This is dependent on the web server. To avoid re-entering this data each time, the Easy Authenticator can capture this “profile” information once and provide the data to fill registration forms. The user will always have a chance to validate, edit, or remove information before sending it over to the web server. Registration is never automated. See Figure 6 for an example of a user profile define in the app.

The user profile interface allows the user to create as many different profiles as needed. The toolbar can be used to view, edit, clone, and delete profiles. Profiles can be exported to an *XML (eXtensible Markup Language)* file for safekeeping or to transfer them to another installation of the app.



The screenshot shows a 'User Profiles' dialog box. At the top, there's a toolbar with icons for navigation and actions. A dropdown menu is open, showing a list of profiles, with 'John Smith' selected. Below this is the 'Profile Parameters' section, which contains several input fields: 'Avatar' (with a person icon), 'First Name' (filled with 'John'), 'Last Name' (filled with 'Smith'), 'Email' (filled with 'john.smith@easyauthenticator.com'), 'Phone Number' (filled with '555-1234'), 'Address' (empty), 'Title' (empty), and 'Organization' (empty). At the bottom of the dialog are 'OK' and 'Cancel' buttons.

Figure 6: User Profiles Interface

2. Account Management

The goal of the Easy Authenticator app is to facilitate and secure the use of online accounts. The accounts will be stored in a vault file and protected with access controls. This section explains how to perform the interactions with a compatible web server for the creation, management, and ultimately destruction of user accounts.

2.1 Easy Authenticator Tokens

Easy Authenticator maintains a separation between the web servers and itself. The interaction is never fully automated. The web server will encapsulate communications and make them available from a web page. To retrieve the message, the user must grab the token containing the message, drag and drop it in the Easy Authenticator interface. The app will read and analyze the message and assist in preparing the needed response for the interaction to take place.

2.2 Server Interactions

The protocol through which Easy Authenticator talks to web servers recognizes seven distinct interactions between clients and web servers. These interactions will start with a message from the server that will have to be dragged and dropped into the Easy Authenticator app. The client response, which will be defined in the application is dragged back and dropped onto the web page of the server when ready. No interaction is automated to increase the security of the operations and give full control to the user.

The following are the interactions that are recognized by the Easy Authenticator app:

- Account Registration: defining and registering a new account on a web server.
- Signing in: starting a new user session on the web server using an existing account.
- Editing the user account/profile: modifying details of the account stored on the web server.
- Renewing the public key: providing an updated public key to the web server. This is analogous to changing the password protecting an account in password-protected accounts.
- Digitally signing an operation: validating an action by providing a digital signature to confirm it.
- Resetting the public key: this operation is accomplished, if the web server allows for it, so that a forgotten key, or a key that is no longer available can be reset and the user can keep using the account.
- Terminating the account: this is an exchange between the web server and the client by which the user expresses the intention of ending the service and invalidating the account.

2.3 Register an Account

The first basic interaction that needs to occur is the registration of the account on the web server. Whether it is accomplished by the user or by an authorized proxy, this will need to happen before any other interaction can occur.

The operation consists of a back-and-forth negotiation with the web server. By dragging the token to the Easy Authenticator app, it can analyze what the server is providing (identification) and what the server is requesting, i.e., the parameters required to build an account. This will consist of typical information such as an email address, first and last names, and non-typical information depending on the requirements that the server imposes to build the account. For instance, a school could require a student ID number when creating an account.

Of course, the interface provided for the exchange will be at the discretion of the web server. Figure 7 provides an example of what such an interface could look like. The authenticator token (see Figure 8) is the vehicle to transport the registration form to the Easy Authenticator app.

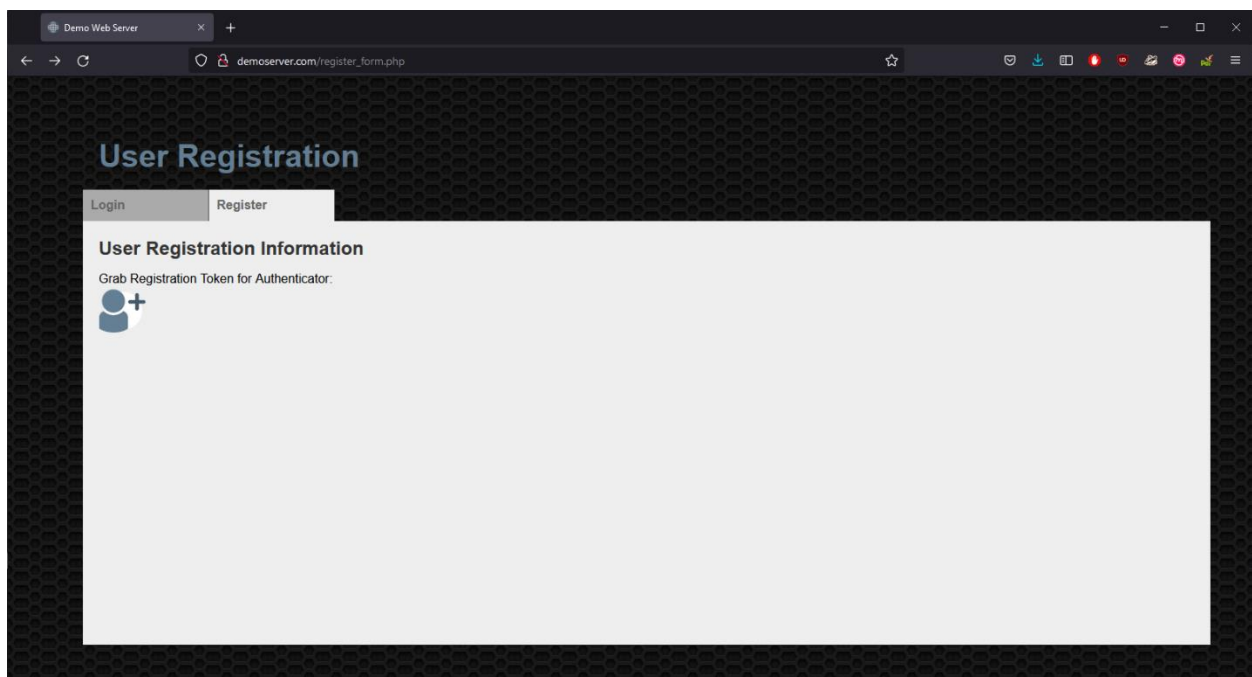


Figure 7: Example of User Registration Interface



Figure 8: Authenticator Token for Registration

Once the token has been dragged and dropped into the Easy Authenticator app, the message is analyzed, and a dynamic form is presented to the user. The form displays the required and optional fields to fill. At the top of the form is the server identification: the name, icon, and URL of the server. See Figure 9 for an example of such an identification.

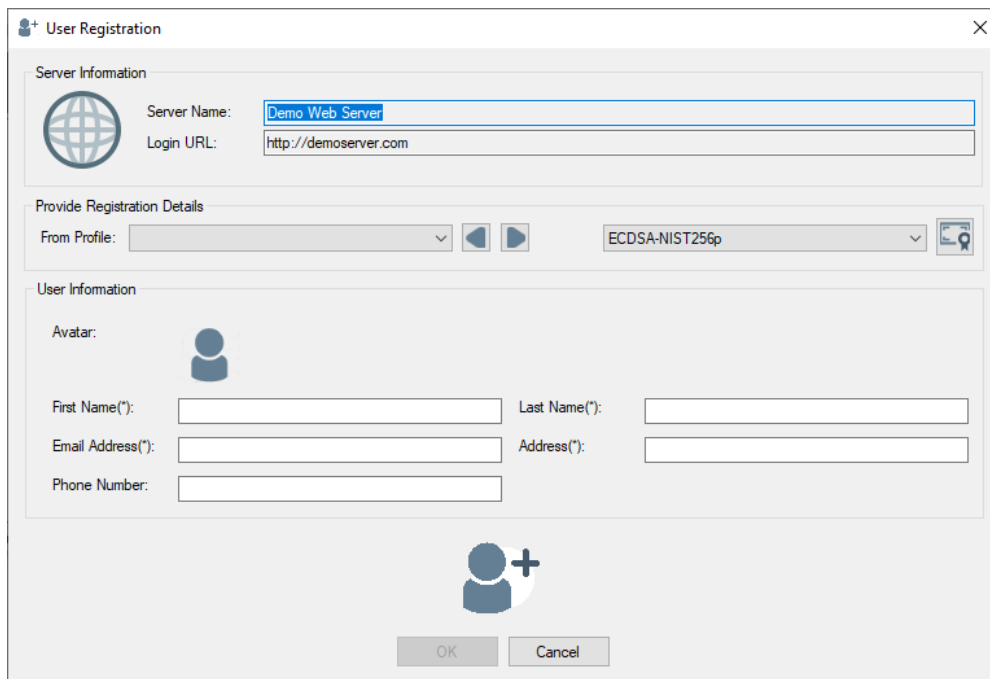
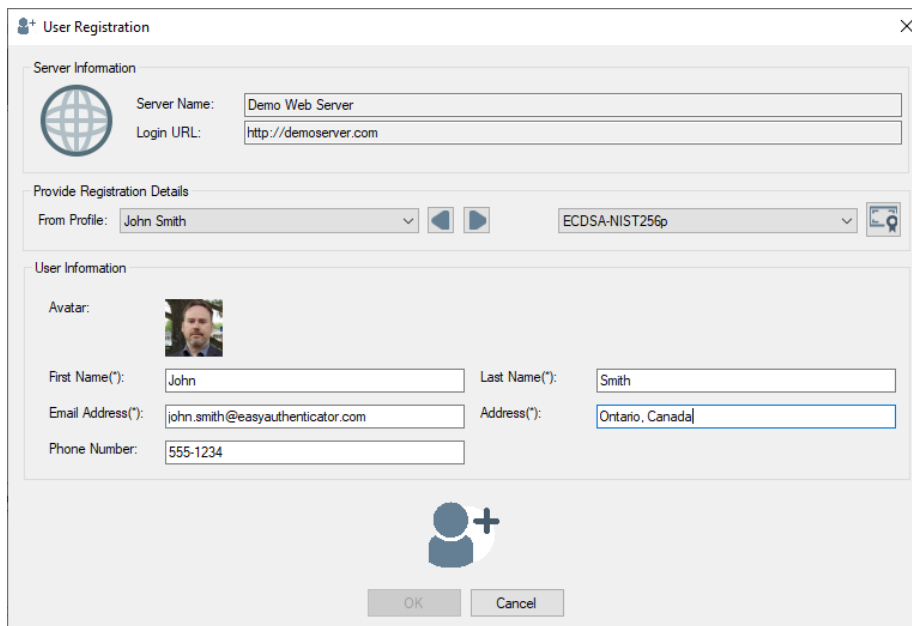
A screenshot of a 'User Registration' dialog box. The dialog has a title bar with a close button. It is divided into three main sections. The first section, 'Server Information', contains a globe icon and two text fields: 'Server Name' with the value 'Demo Web Server' and 'Login URL' with the value 'http://demoserver.com'. The second section, 'Provide Registration Details', features a 'From Profile:' dropdown menu, navigation arrows, and a dropdown for 'ECDSA-NIST256p' with a key icon. The third section, 'User Information', includes an 'Avatar:' label with a person icon, and four text fields: 'First Name(*)', 'Last Name(*)', 'Email Address(*)', and 'Address(*)'. Below these is a 'Phone Number:' label and a text field. At the bottom center is a large blue icon of a person with a plus sign, and at the bottom right are 'OK' and 'Cancel' buttons.

Figure 9: Blank Registration Form

If profiles have been defined in the settings of the Easy Authenticator app, they can be invoked to instantly fill out the form. The exchange protocol defines several keywords that are associated to the profile, and by using these keywords the profile can link up the stored information. If required information is missing or some information needs to be edited, the user can modify it in the form. The form is not automatically sent to the web server in any case: it will only be submitted through a drag-and-drop operation.

2.3.1 Using Auto-Generated Digital Signature Keys

The goal of the Easy Authenticator app is to remove the use of passwords by replacing them with digital signatures. The registration process needs to provide the server with a public key: in the example seen in Figure 9, the key is generated by the authenticator, using the algorithm specified (Elliptic Curve Digital Signature Algorithm using NIST curve 256p). The authenticator will manage the private and public keys and store them in the vault, while a copy of the public key goes to the web server to complete the registration.



User Registration


Server Information

Server Name: Demo Web Server
 Login URL: http://demoserver.com


Provide Registration Details

From Profile: John Smith ECDSA-NIST256p

User Information

Avatar: 

First Name(*): John Last Name(*): Smith
 Email Address(*): john.smith@easyauthenticator.com Address(*): Ontario, Canada
 Phone Number: 555-1234



OK Cancel

Figure 10: Filled Registration Form

To complete the registration process, the form must be submitted to the web server, and the form data must be accepted as valid. The submission is done by dragging the authenticator token that holds the registration information back to the web server.

The web server will validate the account using which ever means it deems acceptable. For example, sending a validation email to the account of the newly registered user to prove ownership of the account is likely to be sufficient in many cases, similarly to password-protected accounts. The user can save the newly created account in the authenticator.

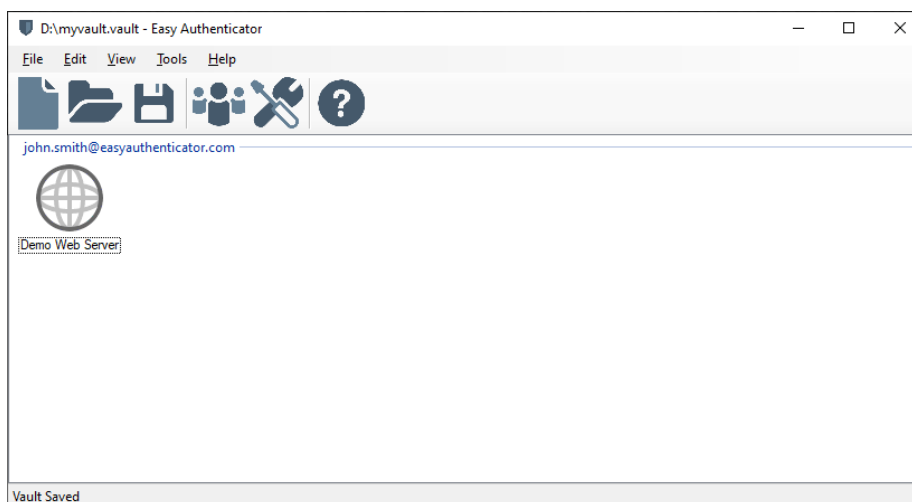
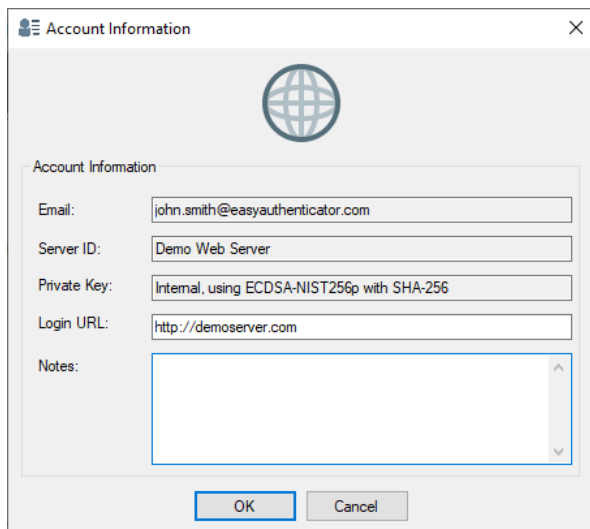


Figure 11: Registered Account in the Vault



Account Information

Email: john.smith@easyauthenticator.com

Server ID: Demo Web Server

Private Key: Internal, using ECDSA-NIST256p with SHA-256

Login URL: http://demoserver.com

Notes:

OK Cancel

Figure 12: Display of Account Details

2.3.2 Using a Certificate

As seen in [section 2.3.1](#), the registration process can leverage the authenticator to create a pair of encryption keys especially for the account. However, it is also possible to refer to an existing X509 digital certificate that contains the public key to use for the account. The same certificate can be used for several accounts since the public key is not sensitive.

When using a certificate to register the account, the user clicks the *Certificate* button on the registration form (see Figure 10) to select the certificate, then provides the password that protects the file, and finally loads the information. This will provide the public key for the registration process. The rest of the information must be provided separately, either from a stored profile or by manually entering the details.

2.3.3 Using a Signed Certificate

In some cases, the web server will ask for a certificate that holds the user's public key, and that is validated by the signature of a Certification Authority, or CA. This provides authenticity to the registration process and validates that the information was verified by a registration authority, or RA. See Figure 13 for an example of such a registration form.

In the message from the web server, the form displays a certificate field, flagged as mandatory. The description of the field states that a certificate, signed by a specific CA, must be provided. The user will be authenticated to that digital identity if they are in possession of the corresponding private key.

This process is like the previously detailed operation to register the account. The difference is that instead of trusting the information provided by the user, the server instead will place its trust in the CA that signed the certificate.

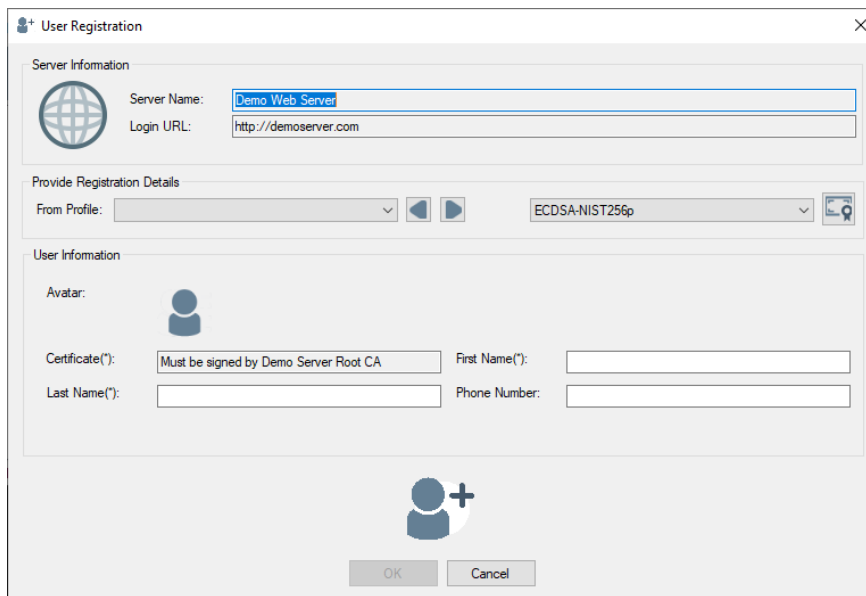
A screenshot of a 'User Registration' dialog box. The dialog has a title bar with a close button. It is divided into three main sections: 'Server Information', 'Provide Registration Details', and 'User Information'. The 'Server Information' section contains a globe icon, a 'Server Name' field with the text 'Demo Web Server', and a 'Login URL' field with the text 'http://demoserver.com'. The 'Provide Registration Details' section has a 'From Profile' dropdown menu, navigation arrows, and a 'Signature' dropdown menu set to 'ECDSA-NIST256p'. The 'User Information' section includes an 'Avatar' field with a person icon, a 'Certificate(*)' field with the text 'Must be signed by Demo Server Root CA', a 'First Name(*)' field, a 'Last Name(*)' field, and a 'Phone Number' field. At the bottom, there is a large blue plus icon and 'OK' and 'Cancel' buttons.

Figure 13: Registration Requiring a Signed Certificate

2.4 Sign into an Account

When the account is registered and validated, the user can now sign into the account and begin an authenticated session. The sign-in interface provides an authenticator token to start the operation (see Figure 15). The operation is like the registration process and all the other interactions with the web server: the token is dragged onto the authenticator app to be analyzed and to prepare a response.

When dragging the token to the authenticator app, the server identification will be read from the message, and the corresponding account will be identified, if it exists. Then, using the private key of the account, Easy Authenticator will generate a digital signature on the challenge provided by the server (see Figure 16).

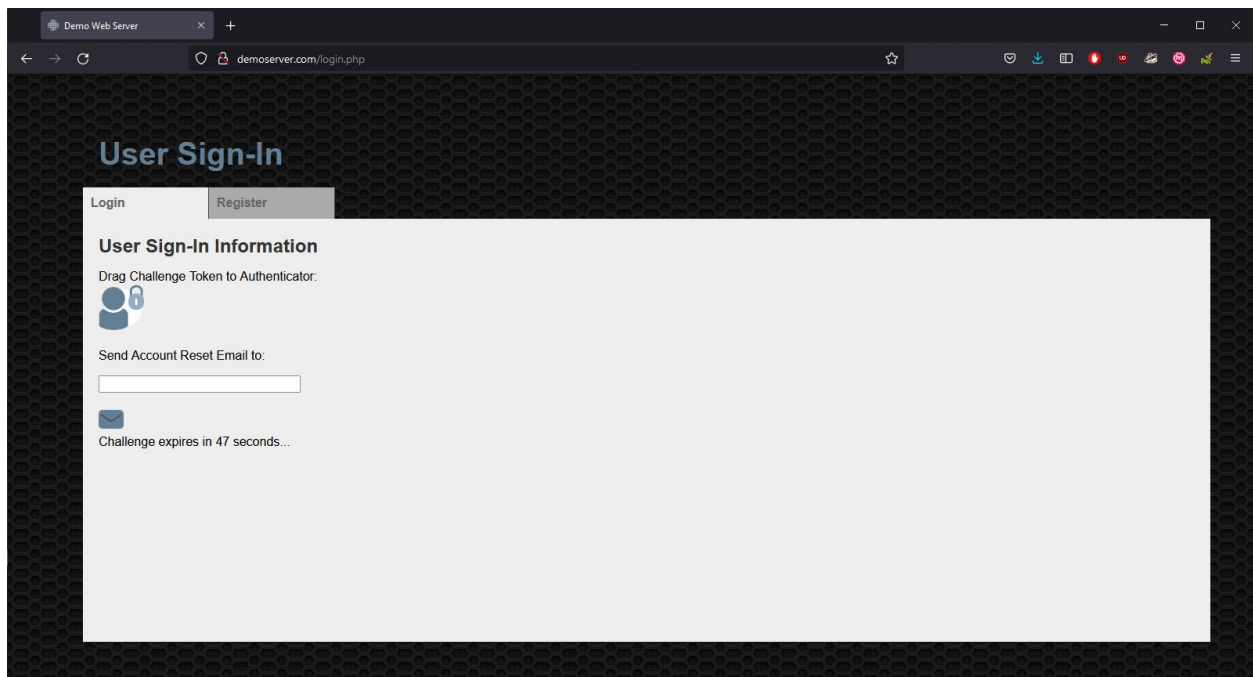


Figure 14: Sign-In Interface



Figure 15: Sign-In Authenticator Token

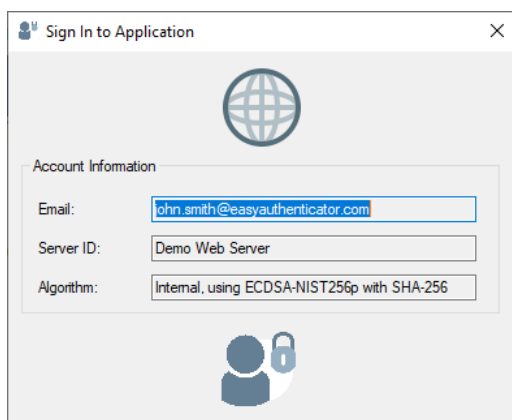


Figure 16: Sign-In Authenticator Interface

The sign-in procedure will display the server identification and a description of the private key used for the signature. If the details correspond to the operation being performed, the user should then drag back the authenticator token from this interface onto the web server.

2.5 Editing an Account on the Web Server (Except the Public Key)

During the lifetime of the account, certain details may need to be edited, such as the phone number, the address, the job title, or even the email address. This operation only concerns the details of the user account that are not used for the digital signatures. To request an account modification, the user will typically access the profile interface of the web server and drag the authenticator token for profile modification to the Easy Authenticator app. See Figure 17 for an example of such an interface.

The authenticator token for account modification (see Figure 19) will hold the current details of the account. Once in the authenticator, the user can modify them, possibly by using a profile. See Figure 18 for an example of the account modification interface.

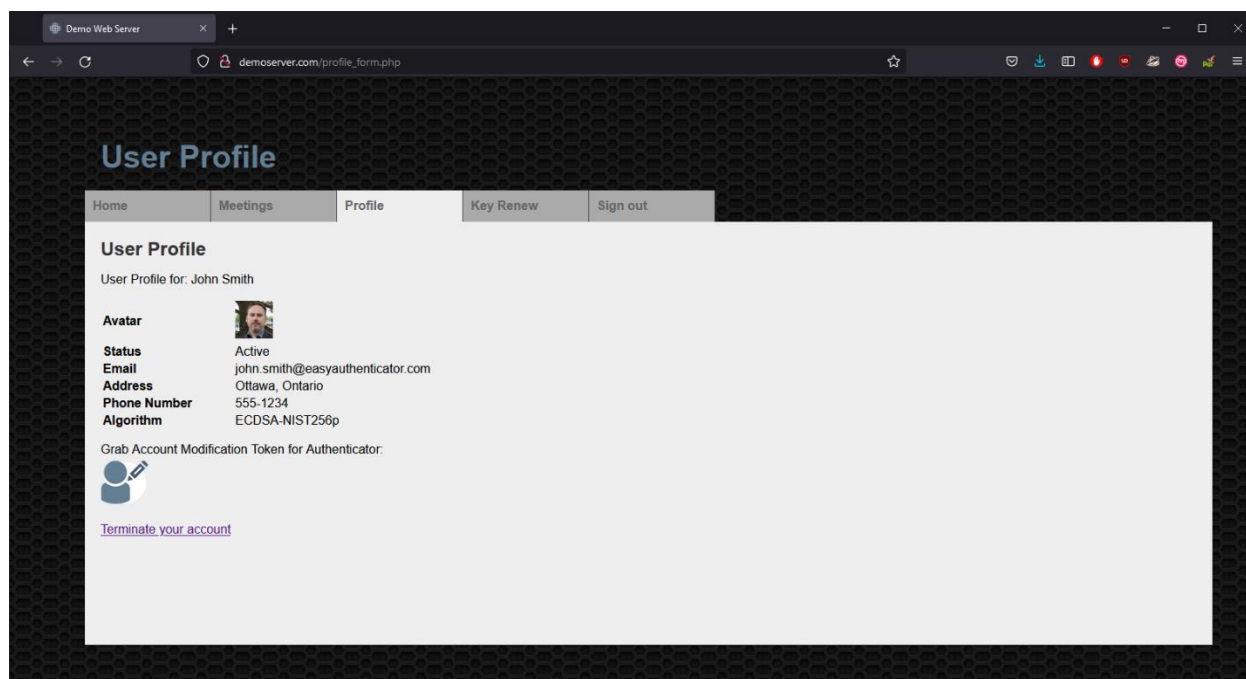


Figure 17: User Profile Interface

Figure 18: Account Modification Interface



Figure 19: Account Modification Authenticator Token

2.6 Renewing the Account's Public Key

The user's public key is stored on the web server to validate the user's digital signature. The public key must correspond to the user's private key, securely detained by the user. If for some reason, the public key needs to be modified, then the user must request a key renewing operation to the web server. Reasons for this might include the digital signature algorithm becoming deprecated. It could also happen if there is some concern that the key might be exposed, through theft or loss of hardware for instance.

The interface that allows the user to renew a key renew is available after the user is authenticated. This means that a valid key must be in possession of the user to do such an operation. If the user does not possess such a key, then the operation required is a [public key reset](#).

When the authenticator token for a key renewal (see Figure 21) is dragged into the Easy Authenticator interface, the application will identify the server through the information provided in the message, and digitally sign the response that will validate the user and provide a new public key. The operation can leverage a certificate, self-signed or CA-signed, as a vehicle for the new public key provided to the server.

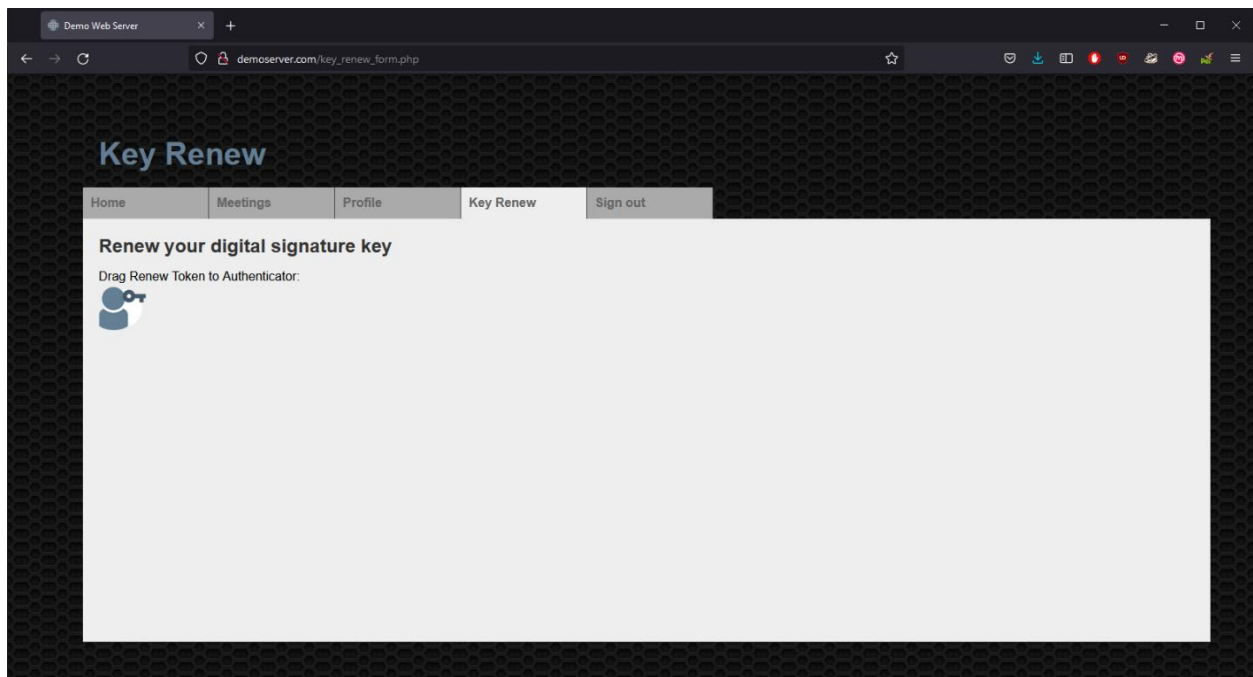


Figure 20: Example of a Key Renewal Interface



Figure 21: Public Key Renew Authenticator Token

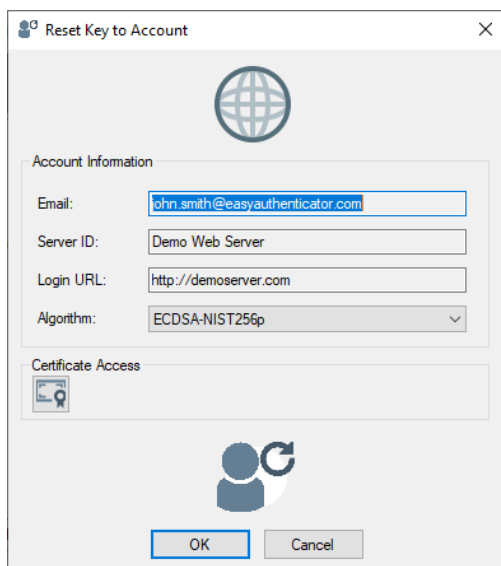


Figure 22: Public Key Renew

2.7 Digitally Sign a Privileged Operation

Digital signatures are a cryptographic tool that provides authenticity to transactions. It can be used in many different contexts, including authentication, but the basic principle is to validate information by encrypting the data with a private key. A privileged operation, one that holds a certain sensitivity, can be specifically signed to protect its integrity and authenticity. This could be described as a generic sensitive transaction from the client to the web server. As with other transaction, the authenticity is validated by a digital signature, with a back-and-forth drag-and-drop operation.

The authenticator token for generic digital signature (see Figure 24) will contain the data to sign, and the authenticator app will take care of the rest. An example of an interface providing a sensitive operation is provided in Figure 23. In the transaction, the web server will encode what is being validated through a digital signature (see Figure 25).

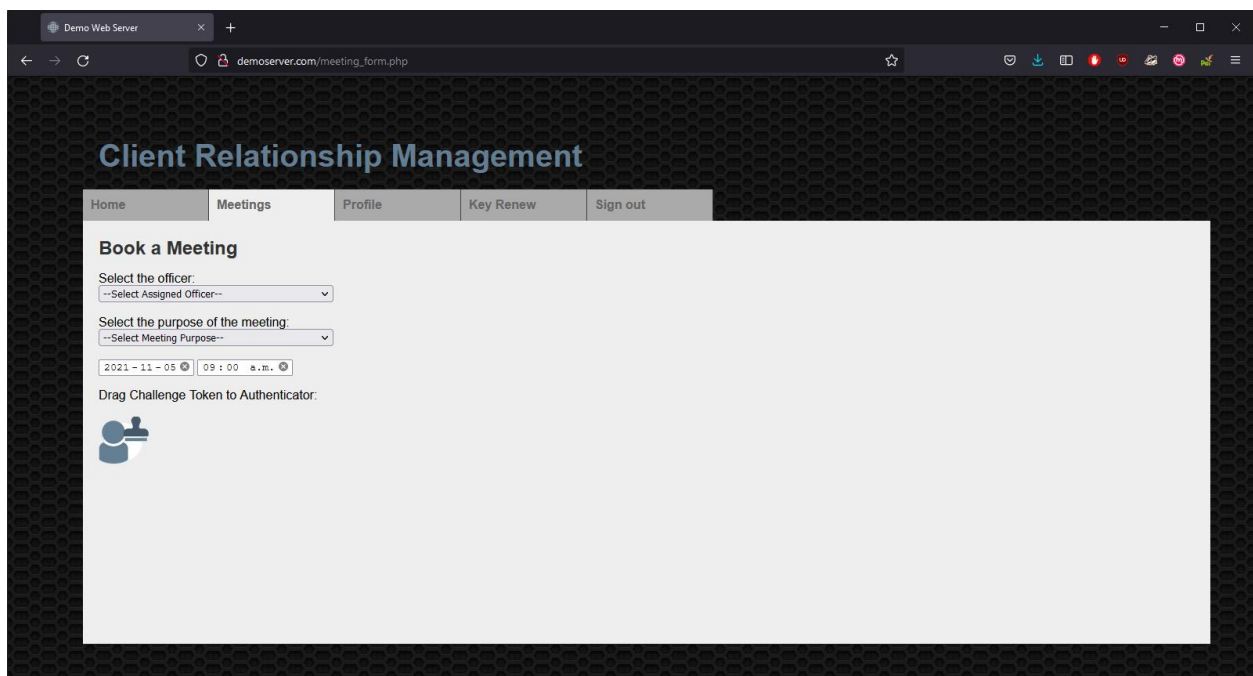


Figure 23: Digitally Signing a Privileged Operation



Figure 24: Digital Signature Authenticator Token

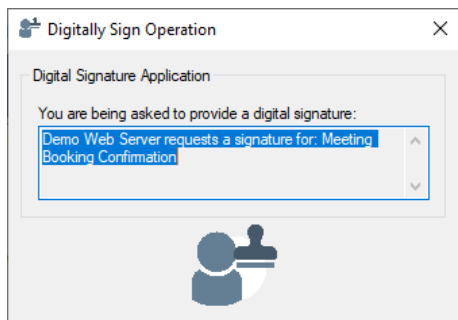


Figure 25: Digitally Signing a Generic Operation

2.8 Resetting a Public Key

An account reset operation is accomplished when the private key is no longer available. This could happen if the vault is lost, if an access control that protects the vault is no longer, or any other reason that prevents the user from opening an authenticated web session.

The process of resetting an account comes with risk. Threat actors could attempt to hijack an account by abusing the account reset procedure. Web service providers need to assess the risk and provide an account reset methodology that mitigates undue risk. This could involve using an email or a phone number associated with the account to initiate a reset or enquiring to customer tech support to resolve the issue. Some service providers might use an associated peer account to validate the operation.

2.9 Terminating an Account

A specific type of interaction between the user and the web server is the request the termination of the user's account. This signals an intention to end the client engagement and invalidate the account on the server side. It is a privileged operation and needs to be digitally signed to protect the integrity of the account.

The user must be within an authenticated session to request an account termination. The procedure starts by dragging an account termination token (see Figure 27) to the authenticator app. The app will identify the account with the information embedded into the token. The user will confirm the intent by dragging the token back to the web server. The exact nature of an account termination is dependent on the web service provider. On the client side, the authenticator app will delete the account from the vault, as it should no longer be usable.

An example of an account termination interface is provided in Figure 26, and an example of the Easy Authenticator interface for account termination can be seen in Figure 28.

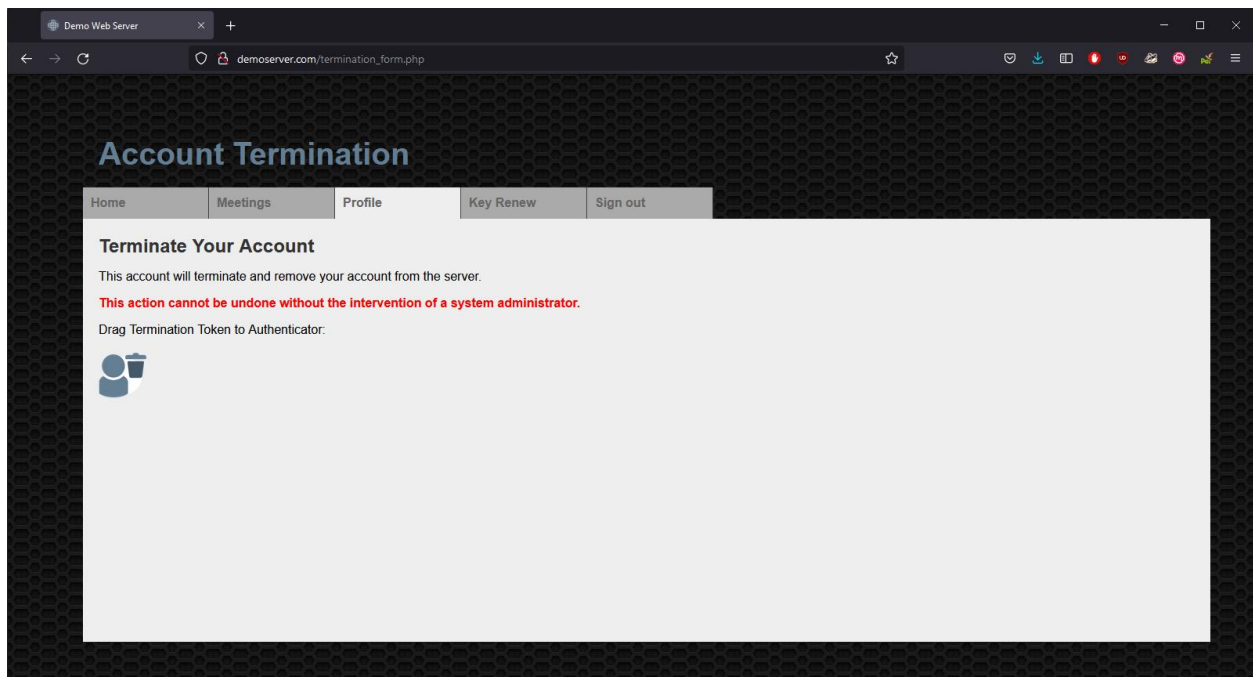


Figure 26: Account Termination Interface



Figure 27: Authenticator Token for Account Termination

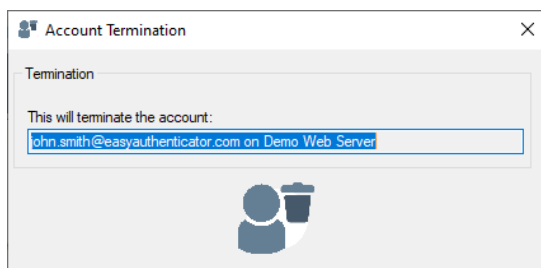


Figure 28: Easy Authenticator App Account Termination Interface

2.10 Account Deletion

Note that it is possible to delete an account from the vault directly, using the Easy Authenticator interface. This does not terminate the account on the related web server. No information will be relayed to the web server from this operation. Furthermore, deleting the account directly will make it impossible to sign in using the account and performing an account termination procedure on the web server. An account should only be directly deleted from the vault once it is already invalidated on the server side.



Warning: Directly deleting the account removes it from the vault but not the web server. To access the account again, it will likely need to be reset.

3. Easy Authenticator Tools

3.1 Yubikey Integration

Easy Authenticator integrates several uses of a Yubikey. It can be used to secure access to vault files, generate and store private and public keys, and certificates.

To access the basic settings of a connected Yubikey, the user must use the interface at *Edit / Yubikey Settings...*

Figure 29: Yubikey Settings Interface

3.1.1 Static Password with the Yubikey

Passwords can be used to protect vault files, but they are not convenient to generate and remember. User can be tempted to use small and easy to remember passwords which diminishes the protection afforded. A solution for this is to use the static password feature of a Yubikey to store, remember, and type the static password. Writing the password becomes as simple as placing a finger on the connected Yubikey.

There are two slots on the Yubikey reserved for either a static password or a challenge-response: slot 1 (short press) and slot 2 (long press). The interface (see Figure 29) allows the user to set a static password. It can be set by the user, or randomly generated by the Yubikey.

When choosing to have the Yubikey generate the password, select the keyboard configuration to use. Because the Yubikey emulates keystrokes on the keyboard, the password generated may be affected by the keyboard configuration. Yubico has developed the modified hexadecimal keyboard (ModHex) to generate passwords that are consistent on any keyboard configuration. This limits the characters that can be used and diminishes the overall entropy, but this can be compensated with longer passwords. If the keyboard configuration used is always going to be English US, this can be selected for the password generation.

To test the password, simply place the focus on an editable entry form (like in a basic text editor) and press a finger to the key. The password (in clear) will be typed in the entry form.



Warning: Loss or failure of the Yubikey could mean that the password is no longer retrievable. It is recommended to keep a copy of the password in a safe location.

3.1.2 Challenge-Response with the Yubikey

The challenge-response feature can be used to protect access to a vault file. There are two components involved in this operation: the private key and the challenge submitted. Because the encryption and decryption processes need to generate a specific key, the same private key and challenge must be used consistently to access the file. The private key will be stored on the Yubikey (in slot 1 or 2) and the challenge will be stored in the settings of the Easy Authenticator app. A threat actor trying to decrypt a vault file protected with the scheme will need both pieces of information to be successful.

To configure the parameters in the Easy Authenticator settings, In the Yubikey settings interface (see Figure 29), the user should first select the device, then the slot that will be configured for the challenge-response operation. In the section “Set Authenticator Challenge”, the user can define which challenge to submit to the Yubikey and to which slot. Those settings will be used when creating a Yubikey access control.



Warning: To have the capacity to recreate those settings (on another Yubikey for instance), you must keep the private key and the challenge in a safe location, preferably offline.

3.1.3 Generating a Self-Signed Certificate on the Yubikey

A Yubikey has a total of 25 slots that can contain Personal Identification Verification (PIV). While some are reserved for specific uses, several can be leveraged by the user for authentication and digital signatures. Easy Authenticator can interact with the Yubikey to generate pairs of private/public keys to be stored in PIV slots. The private key is then securely stored on the hardware device and is not in the associated vault. The vault will only contain tracking information: the Yubikey serial number and the slot where the key is located. This allows a quick and easy interaction through Easy Authenticator when managing accounts. Several accounts can be protected through the same digital identity.

A Yubikey can store a private key and its corresponding public key in a slot. The public key is embedded in a certificate for standardized manipulation of the data. Easy authenticator can communicate with a Yubikey to generate such information and store it on the Yubikey.

To access this functionality, use *Tools / Yubikey PIV Editor...* An example is provided in Figure 30.

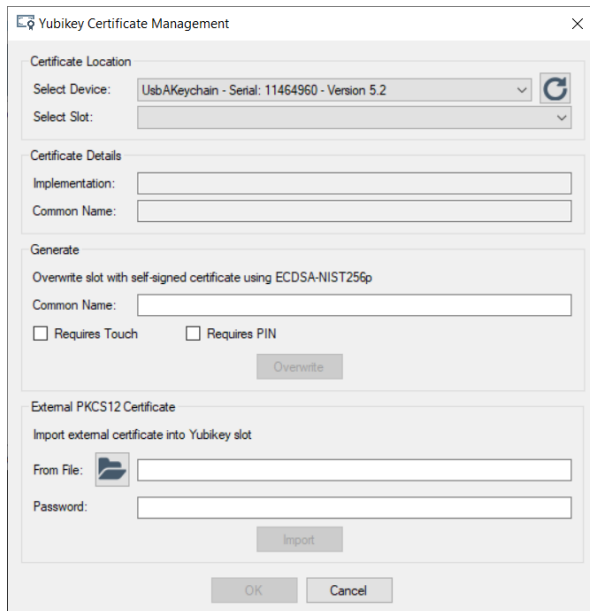


Figure 30: The Yubikey PIV Editor Interface

To get started, the Yubikey must first be selected at the top of the interface. If the key does not appear, or is connected after the interface has been generated, the *Refresh* button to the right can be used to actualize the list of devices. Once the key is selected, the slot that will be edited must be selected. It is possible to overwrite a slot that already contains information or to put information into an empty slot. When overwriting a slot, the information it previously contained will not longer be available and accounts requiring the private key that was stored at that location will become unreachable.

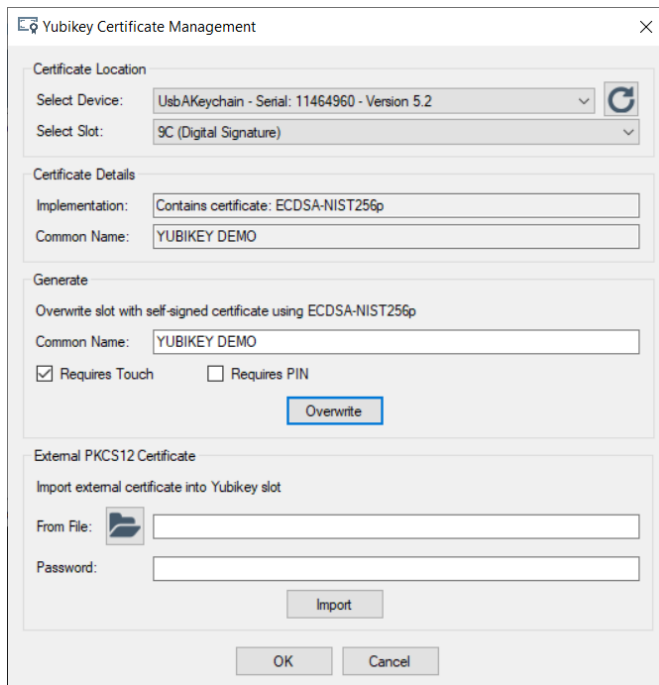


Figure 31: Generating a self-signed Certificate on a Yubikey

3.1.4 Importing a Certificate on the Yubikey

The bottom portion of the interface is to load an existing certificate into a Yubikey Slot. If you are registering an account that is validated through a signed certificate, this certificate can be loaded into the Yubikey for management and protection, instead of being in a password-protected file.

The format expected is PKCS#12, usually having a file extension of **.pfx*, **.pem*, or **.p12*. This format encapsulates the public certificate and the private key in one file. There is usually a password protection on the file, and the password must be provided when importing the certificate. The private key will be stored alongside the certificate in the Yubikey. Afterward, the certificate can be exported but not the private key, so it is recommended to keep a copy of the certificate file in a safe location.

Once imported, associate the account that is being registered with the certificate in the Yubikey, as you would for a [file-based certificate](#).

3.2 USB Security Key

3.2.1 What is a USB Security Key?

A USB Security Key is a hardware storage for a key file, that can be use as a token to access encrypted vault files. The key can be any type of USB key. At the root of the key, there will be two files: one is titled *"key_file.txt"* and is the file from which the encryption key is derived by hashing the contents of the file. The contents do not really matter as long as they are unpredictable and protected from unauthorized access. The second file is named *"security_key_id.txt"* and contains the identification

string of the hardware key. This should be a recognizable identification to distinguish this security key from others.

3.2.2 How to Generate a Key

Although the security keys can be manually created by creating the required files at the root of a USB key, Easy Authenticator provides an interface to easily create a key. The interface is invoked by the menu at *Tools / USB Security Key...*

In Figure 32, you can see an example where Easy Authenticator detected a USB Security Key (by validating the presences of the two files previously described) and has displayed the volume label and identification string of the key. The example uses a fingerprint protected USB key, which can help further secure the contents through biometric authentication.

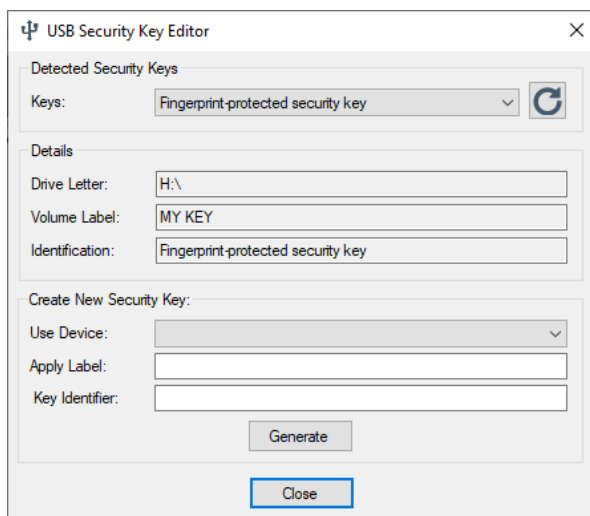




Figure 32: USB Security Key Editor Interface

The bottom part of the interface is used to easily create a new USB Security Key. The drop-down list will display all the removable drives connected to the system. To create a key, the drive must be selected, then optionally the volume label is defined, and the identification string is provided. When clicking *Generate*, the files will be created or replaced at the root of the key. The key file used for encryption will be randomly generated and written to the USB automatically, also overwriting any previously existing key file. No file other than these two specific ones is affected by the operation, and the USB key can contain other content, including vault files.

 **Warning:** If the key is already a USB Security Key, the files will be overwritten, and any vault secured with the key file that was previously on the USB key may become inaccessible.

 **Warning:** USB Keys can be lost, stolen, or break down. It is advisable to keep a copy of the key file in a secure location to be able to recreate the USB security key if needed.

3.2.3 How to Use a USB Security Key to Secure a Vault file

The USB Security Key, or more precisely the key file it contains can be used to derive the encryption key for a vault. The key can be compounded with other security controls for a more robust layer of protection. In the vault access interface, the available USB Security Keys will be detected and can be selected from the drop-down list. Easy Authenticator will use the key file on the selected USB key to generate the security parameter.

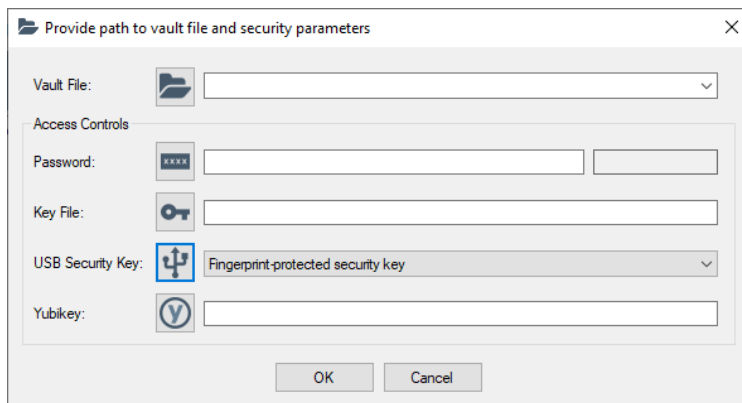


Figure 33: Selected USB Security Key for Vault Access

3.3 Certificate Viewer

As a utility function, Easy Authenticator allows the user to view the contents of a X509 certificate. This feature is invoked from the menu *Tools / Certificate Viewer...* To view the contents of a certificate, the user must provide the path, the password if one is required, and load the details. See the example in Figure 34.

The interface also provides access to certificates that are within a Yubikey. In this case, the user will need to click on the Yubikey icon, select the Yubikey connected to the system that contains the certificate, and select the slot where the certificate is located. The details of the certificate will then be displayed in the interface when the *Load* button is clicked.

Digital Certificate

Import Certificate

From File:

From Yubikey:

Password:

Certificate Details

Format: Version:

Serial Number:

Thumbprint:

Valid From: To:

Holds Private Key: Expires:

Issuer

Name:

Canonical:

Signature Algorithm:

Subject

Name:

Email:

Canonical:

Signature Algorithm:

Figure 34: Details of a Loaded Certificate