

SecureAuth Authentication: SecureAuth performs what was impossible using X.509 certificates

Many enterprises are struggling with the dual conundrum of dealing with increased security concerns and the cost and complexity of solutions. The increased concern in security is triggered from both real security threats like the Kaminsky attack and regulations like PCI, FFIEC, SOX, etc. Classic Public Key Infrastructure (PKI) technology is one of the solutions that is usually eliminated in the planning stages due to cost and complexity. The costs for authentication solutions include initial purchase price as well as deployment and ongoing maintenance costs.

There are some significant differences between traditional PKI and SecureAuth:

1. **PKI is difficult to deploy - SecureAuth is Not**
2. **PKI is difficult to maintain - SecureAuth is Not**
3. **PKI is difficult to scale – SecureAuth is Not**
4. **PKI is difficult for Enterprises to Integrate into Applications – SecureAuth is Not**
5. **PKI is difficult for end users to obtain/transport credentials – SecureAuth is Not**
6. **PKI is only single factor authentication – SecureAuth is 2 Factor**

For these reasons most enterprises have justifiably avoided deploying PKI solutions, despite the fact that public/private key pair technology has proven to mitigate Man-in-the-Middle attacks such as the Kaminsky DNS attack. (See figure 1)

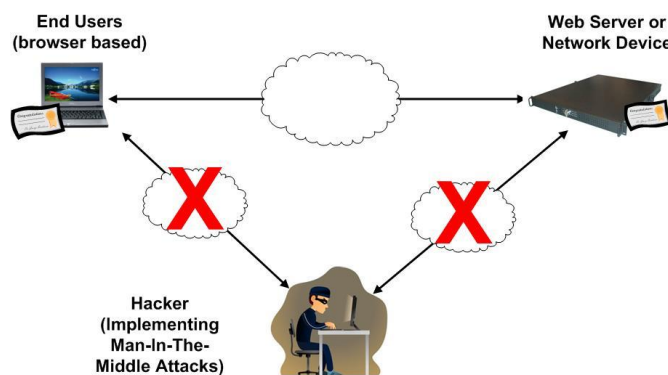


Figure 1 - X.509v3 Technology is proven to prevent man-in-the-middle identity attacks

SecureAuth utilizes X.509v3 technology but avoids all the complexities and cost of PKI. The result is an authentication solution that is deployable, scalable and affordable.

To validate the SecureAuth technology, this paper will enumerate on the above points why “SecureAuth is not PKI” and thus the correct authentication choice.

1. PKI is Difficult to Deploy – SecureAuth® is Not

Classic PKI requires several infrastructure components to function:

- A Redundant Set of Certificate Servers
- A Redundant Set of Certificate Revocation Lists
- A data store of certificates issued
- A set of server for out-of-band registration

In addition, if the “enforcement point” is a web server, then the webserver must be converted to allow Client Side SSL (C-SSL). This is not a trivial task and breaks many pre-existing applications. (See Figure 2)

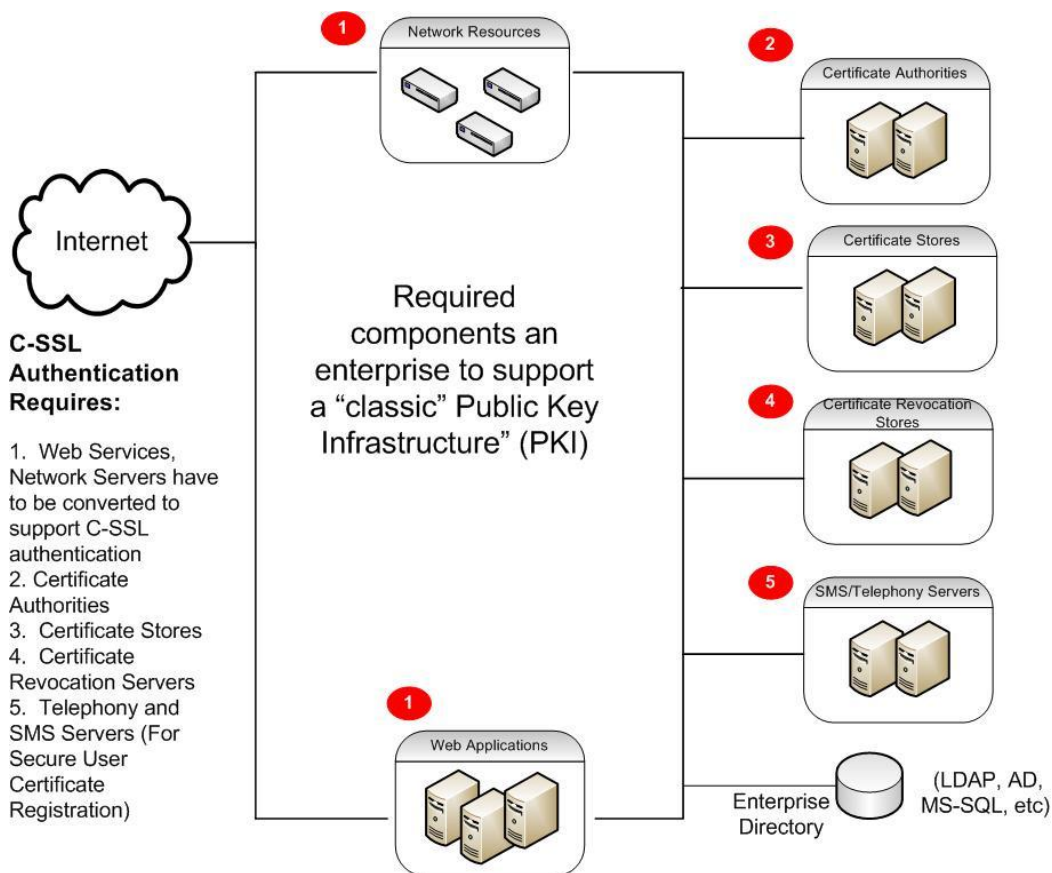


Figure 2 – “Classic” PKI has an onerous amount of infrastructure to install

SecureAuth acknowledges that this infrastructure is burdensome on an enterprise and SecureAuth does NOT require these components for an installation.

1. PKI is Difficult to Deploy – SecureAuth is Not (Cont'd)

SecureAuth installs a simple authentication appliance that meets the requirements of the above components without the overhead. SecureAuth is able to meet these functional requirements through two key architectural differences:

a) SecureAuth utilizes the native data store

The SecureAuth authentication appliance is shipped with data store connectors that enable it to integrate into a native data store. This ability to use the “data store of record” means that there is no data syncing and no need for certificate retention and certificate revocation list. (See Figure 3)

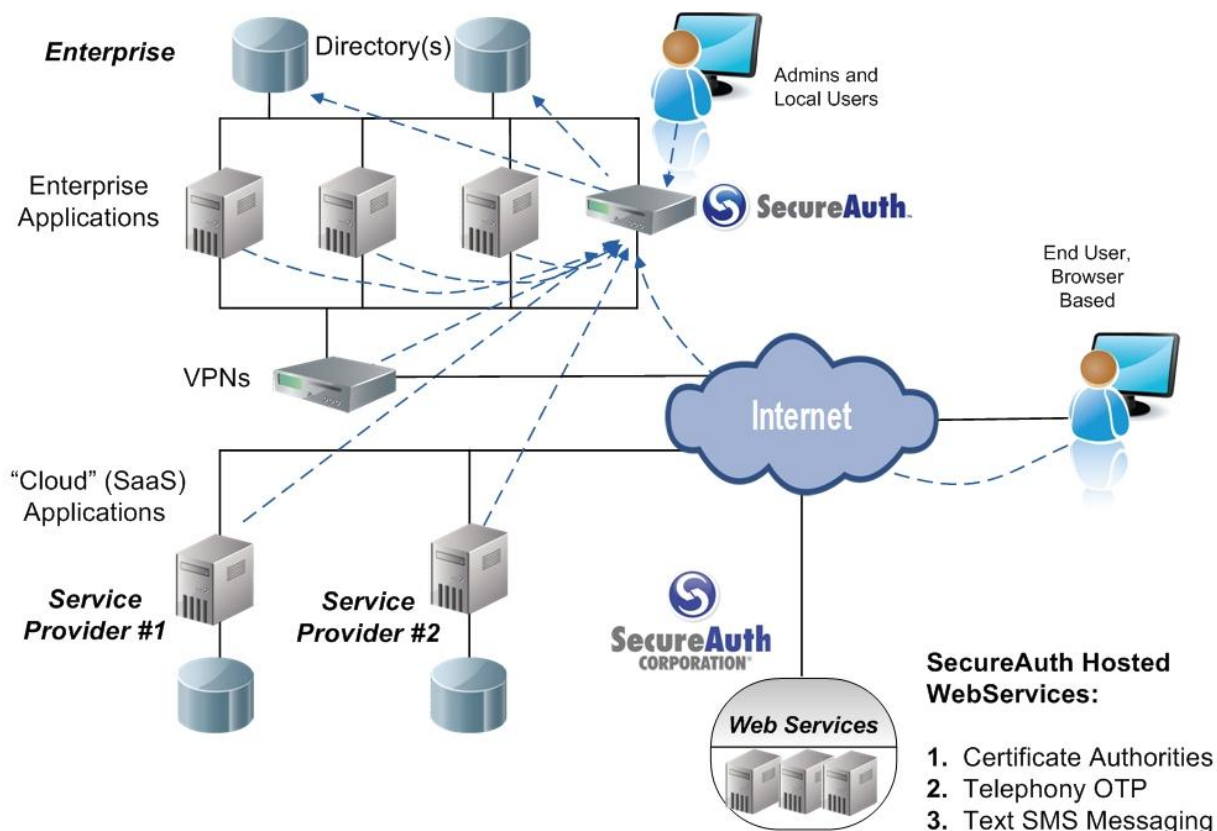


Figure 3 – SecureAuth in a deployed environment for web, network and SaaS authentication.

b) SecureAuth utilizes secure web services

Upon deployment the SecureAuth authentication appliance is enabled to utilize SecureAuth’s hosted webservices for key functional needs. The SecureAuth webservices include:

- Telephony One-Time-Registration Passwords
- SMS One-Time-Registration Passwords
- Certificate Servers

1. PKI is Difficult to Deploy – SecureAuth[®] is Not (Cont'd)

These web services allow for a simple installation and integration of the SecureAuth authentication appliance into a network and web deployment. (See Figure 3)

Summary:

Because standard PKI has so many infrastructure components to install and maintain – it has typically been eliminated as a deployable authentication solution. (See Figure 2) SecureAuth eliminates these deployment issues by:

- Utilizing the existing data store, requiring no data synching
- Simple Appliance installation
- Utilizing webservices for authentication calls

2. PKI is Difficult to maintain – SecureAuth is Not

PKI is the only technology in the world that requires:

- A list of all valid credentials
- A list of all invalid credentials

It is this 2nd list, the “anti-database” that has made PKI all but impossible to maintain.

It’s as if an e-mail administrator were asked to:

- Keep all current valid users
- Keep all users, once valid, but no longer valid

Imagine if an e-mail administrator were asked to put in a system in which each time a user was to gain access, he would require the system to check multiple lists to see whether the user was first, NOT valid. The admin would ask you, “Why would you do this? Why not just check the current data store and check if the user is valid?” No other system of authentication first checks “who is not valid” before determining who is valid.

PKI does this, not for technology reasons but for legacy reasons. The entire PKI infrastructure was invented before directories were a standard resource in enterprises. Hence PKI came up with “anti-database” technology to compensate for this lack of “central data store” of record – to understand who is available.

The dates of these “anti-database” technologies is illuminating:

CRL – Certificate Revocation List ([RFC 1422](#)) 1993
OCSP - Online Certificate Status Protocol ([RFC 2560](#)) 1999

Both of these technologies were devised and implemented before directories were implemented and available as the data store of record.

To compensate for the lack of a central store, enterprises came up with these CRL and OCSP architectures. Both have proven to be unwieldy and undeployable to anything but limited B2E network deployments. There has been no market acceptance of CRLs and OCSP technologies for B2C and web environments because of this complexity.

SecureAuth leverages the advances in present day technology since 1999, namely the central data store that now exist in virtual every enterprise today (See Figure 3). SecureAuth enables an enterprise to set a “Per User Certificate Limit” in the SecureAuth admin console. This is configurable per SecureAuth instance (See Figure #4).

2. PKI is Difficult to maintain – SecureAuth is Not (Cont'd)

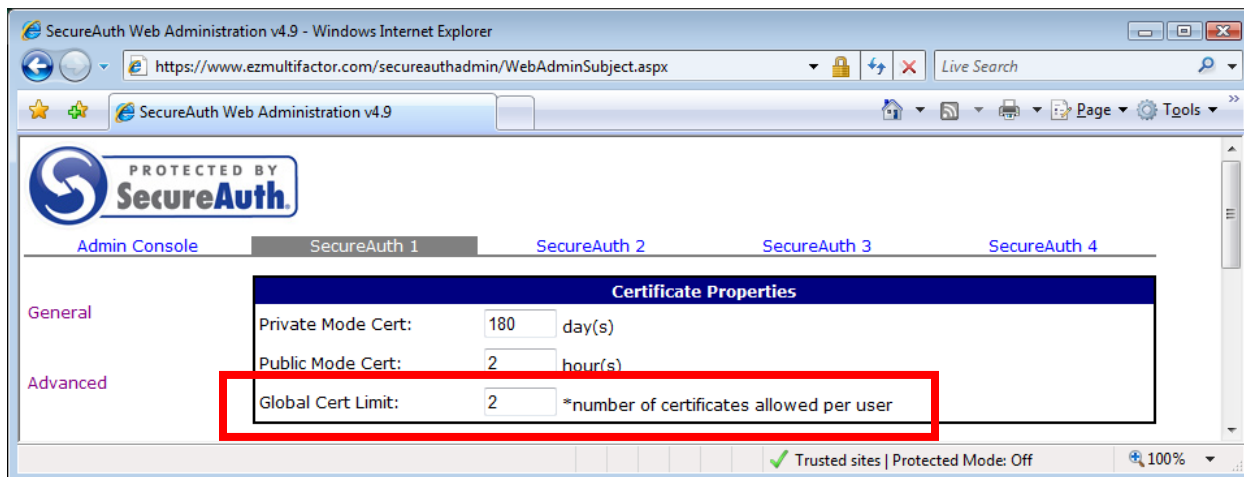


Figure #4 – SecureAuth allows an enterprise to configure # of certificates per user

SecureAuth, because it is an abstracted Authentication Appliance (Figure #3) is able to inspect two sets of information upon every new authentication and credential information. These (2) sets of information are:

- The Instance certificate limit for the resource (Figure #4)
- The Current Certificate Issued , per user (Figure #5, item “A”)

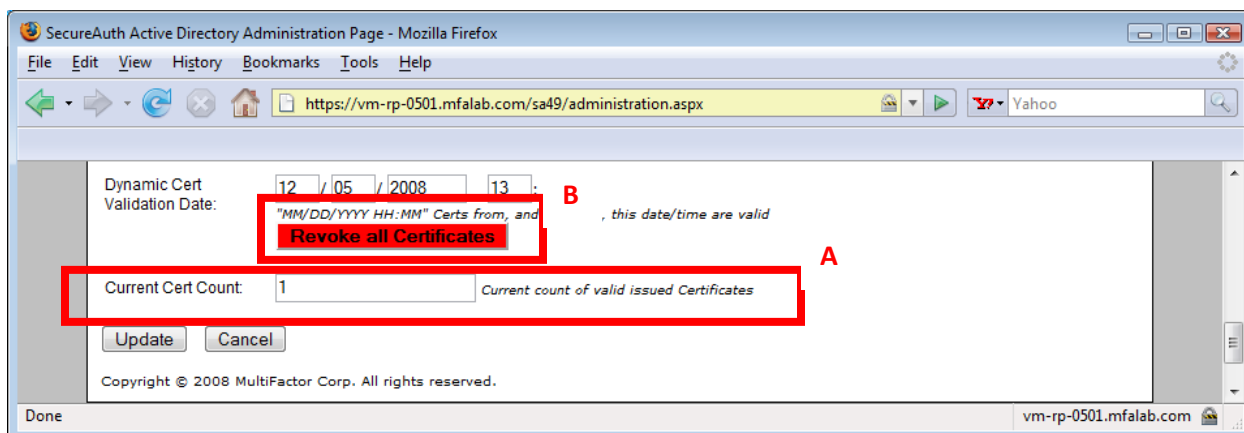


Figure #5 – SecureAuth keeps track of the certificates issued to the user (Item A) and allows “1-touch” revocation with the Cert Revocation Button (Item B).

Because SecureAuth is the authentication enforcement point, SecureAuth is able to inspect current issued certificates and choose NOT to distribute subsequent certificates based on this count.

2. PKI is Difficult to maintain – SecureAuth is Not (Cont'd)

Just as important, a SecureAuth admin is able to revoke the validity of certificates with the action of a single button on the SecureAuth console (See Figure #5, Item "B"). Invoking this feature sets:

- The Validation Date to the current time (See Figure #6, Item "A")
- The Certificate Count to 0 (See Figure #6, Item "B")

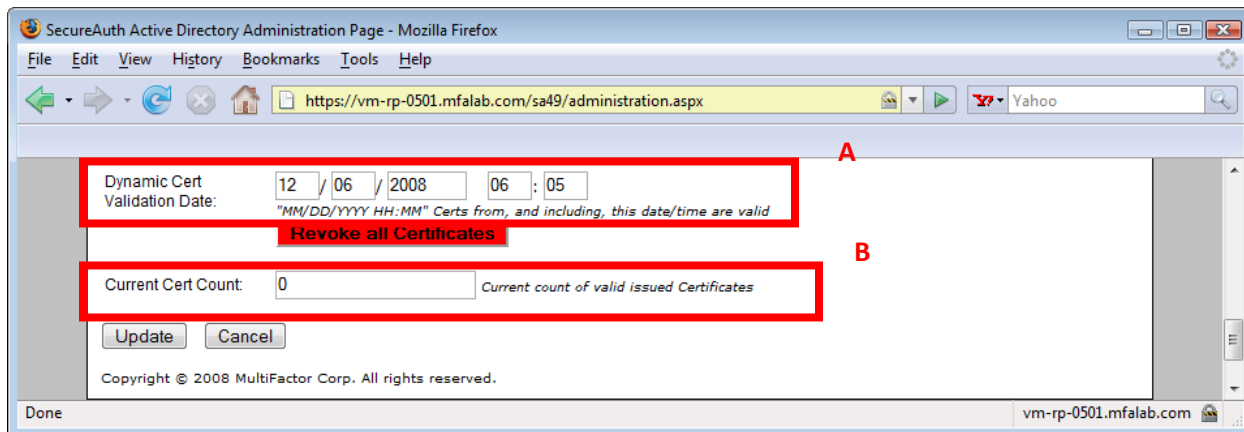


Figure #6 – A single button, “The Revoke All Certificates”, revokes the certificates for the user without CRLs or OCSPs.

The key here, and unique to the SecureAuth solution, is that SecureAuth Authentication appliance is the authentication enforcement point. (See Figure #7) SecureAuth does not have to rely on the “kludgey” and hard to manage CRLs and OCSP infrastructure to enforce the revocation – SecureAuth is the enforcement point and thus can inspect and choose not to accept the credential.

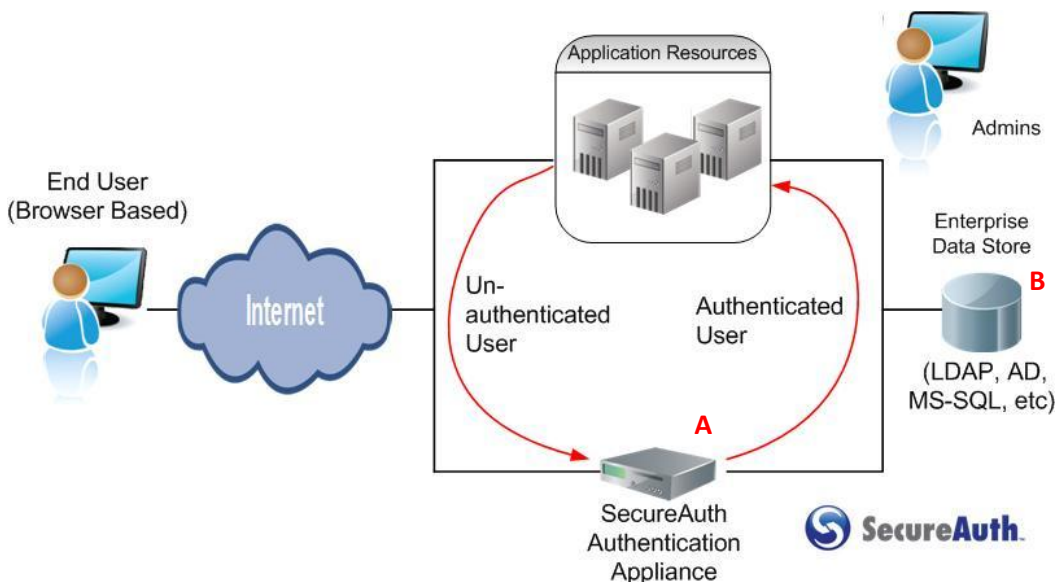


Figure #7 – SecureAuth (Item A) is the enforcement point and thus can check the certificate off the information stored in the central data store (Item B). No CRLs or OCSPs are needed.

3. PKI is Difficult to scale – SecureAuth is Not

Most enterprises are very comfortable with deploying high availability IT resources. The concept of load redundant web servers and load balancers is commonplace in the modern IT infrastructure.

Unfortunately, Certificate Authorities (C.A.s), the core component of the PKI infrastructure, does lend itself to this type of scaling. Unlike web servers, the concept of simply adding another server and the placing a load balancer in front of the set of certificate servers simply will not work. In addition, Certificate Authorities do not have any concept of clustering.

Creating a high availability C.A. infrastructure is not an activity that most IT enterprises have the talent or resources to create. The knowledge required to create a Trusted Root Certificate Authority and then create a set of Intermediate Servers signed with the Trusted Root Certificate, is not common place in today's environment. The cost associated with this skill set is prohibitive for a high availability deployment of PKI to most enterprises.

SecureAuth alleviates this cost set by removing this costly infrastructure from the enterprise. (See Figure #8, "Item B"). A secure WSE 3.0 Web Service call is made between the authentication appliance (Figure #8, Item A) and the SecureAuth Web Service (Figure #8, Item A). Because this call is X.509 v3 authenticated web service connection, and requires the enterprise to have a registered SecureAuth server to access the Web Service.

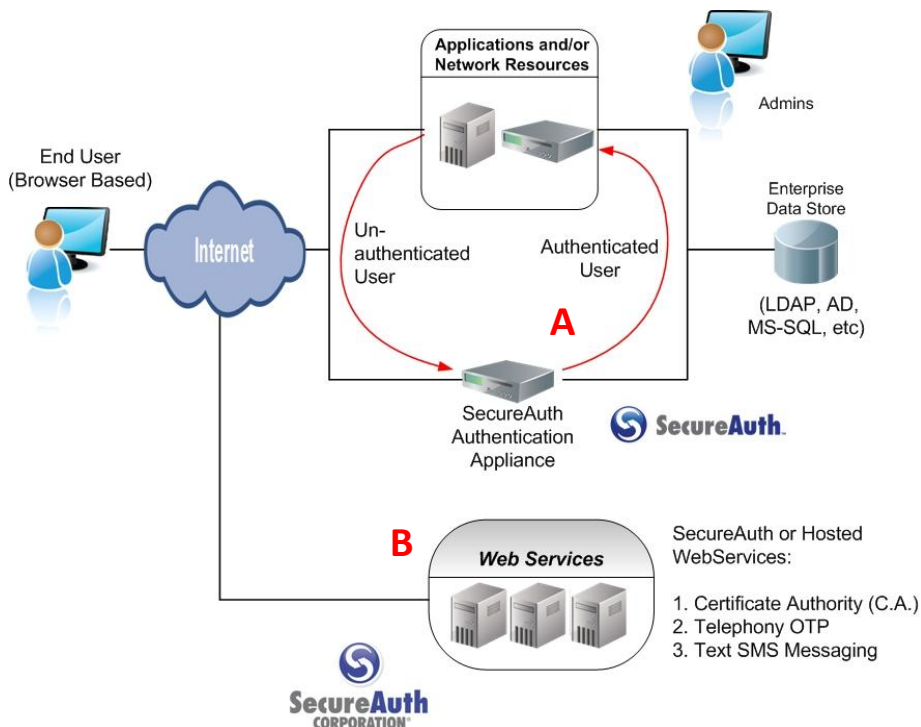


Figure #8 – The authentication appliance (Item A) makes a secure WSE 3.0 authenticated webservice call to the SecureAuth Certificate Authorities (Item B).

3. PKI is Difficult to scale – SecureAuth is Not (Cont'd)

Creating a set of certificate authorities residing behind a protected set of Web Services is not a trivial task and beyond the scope of most enterprises. SecureAuth has created a unique set of high-availability certificate authorities behind a protected set of Web Services. (See Figure #9) This is an industry unique configuration that:

- Securely created/distribute 509v3 credentials without infrastructure
- Scale their authentication to infinite user counts

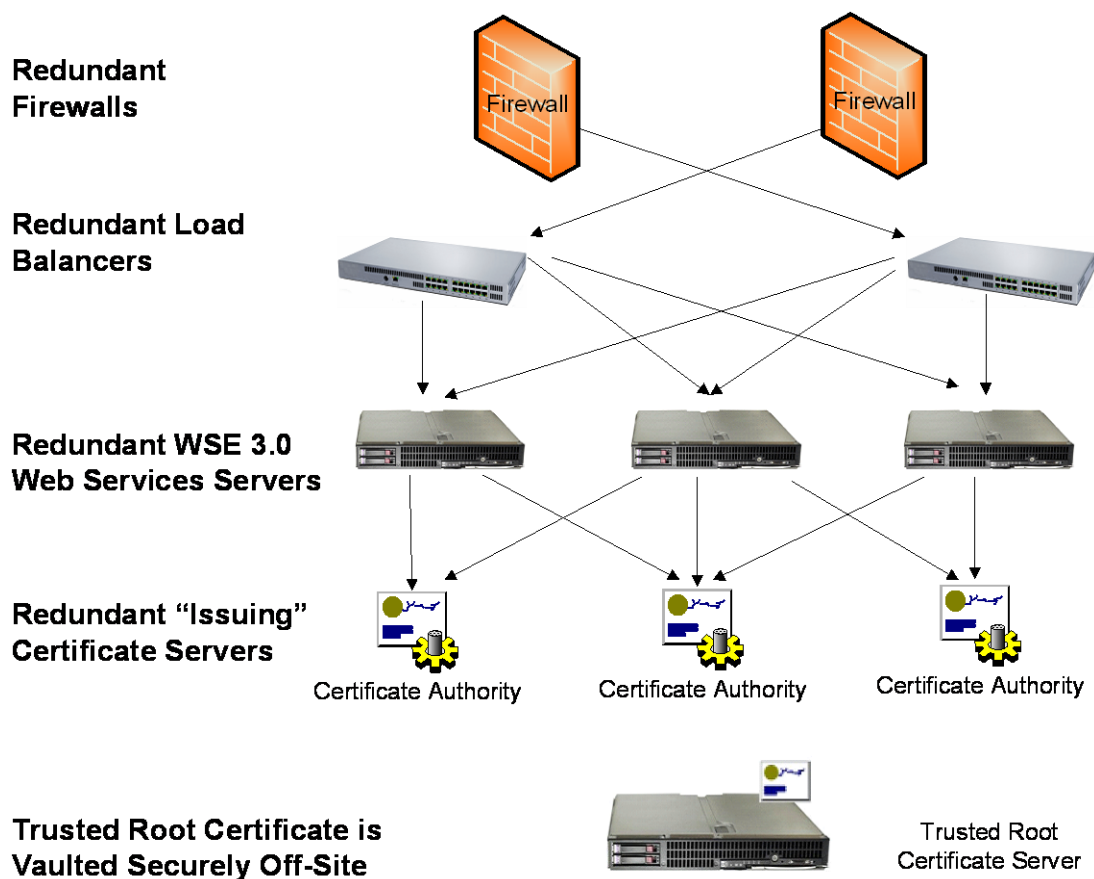


Figure #9 – SecureAuth hosts a high availability infrastructure of certificate authorities authenticated by WSE 3.0 webservices. SecureAuth can also aid the enterprises in creation of this infrastructure, on-premise, for those that choose to host these services

It important to note that different end user X.509 certificates created from this infrastructure contain information ONLY germane to the specific enterprise deploying a SecureAuth authentication appliance. A certificate created for enterprise A is not necessarily valid for enterprise B – despite being issued from the SecureAuth infrastructure. SecureAuth is able to do this via an advanced set of policy servers

3. PKI is Difficult to scale – SecureAuth is Not (Cont'd)

executing policy on the SecureAuth Certificate Authorities. The information is “tagged” in the PKCS #10 request sent from the SecureAuth appliance (Figure 8, Item A) to the SecureAuth Webservice C.A. (Figure 8, Item B). The result is an end user certificate with OU and DC information in the end user certificate that maps only to the deploying SecureAuth authentication site.

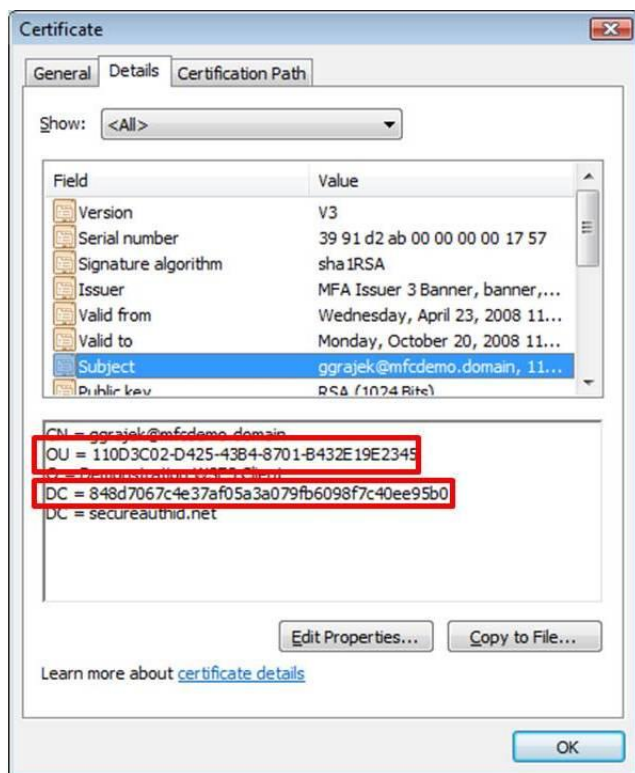


Figure #10 - The enterprises is assigned a unique OU and DC that maps to the SecureAuth authentication appliance registered to the enterprise.

4. PKI is Difficult for Application Integration – SecureAuth is NOT

It is important to note that for PKI deployments, the act of creating a PKI infrastructure and then deploying the certificates does not complete the task for the enterprise – the enterprise now has to determine how to make its current application and network resources accept an X.509 v3 authentication.

PKI for applications requires web services to “turn-on” Client side SSL authentication (C-SSL). This is not a trivial activity (Microsoft Outlook Web Access has a 32-page documentation on the process). It is NOT a matter of just re-writing the authentication page. The actual web server or virtual web server has to be configured to conduct a C-SSL authentication. This is a global setting that has ramifications across all applications that are hosted on the web server. It’s important to note that the process of turning on and supporting C-SSL on web servers is NOT the same across disparate web servers:

- Microsoft IIS
- Apache
- IBM HTTP-D
- Zeus

SecureAuth Solves the Application Integration Issue

Applications do not have to convert the web-server to enable C-SSL authentication to utilize the strength of SecureAuth’s bilateral X.509v3 authentication. Unlike C-SSL, SecureAuth is able to utilize the native target/redirect and sessioning mechanisms of the application (See Figure #11).

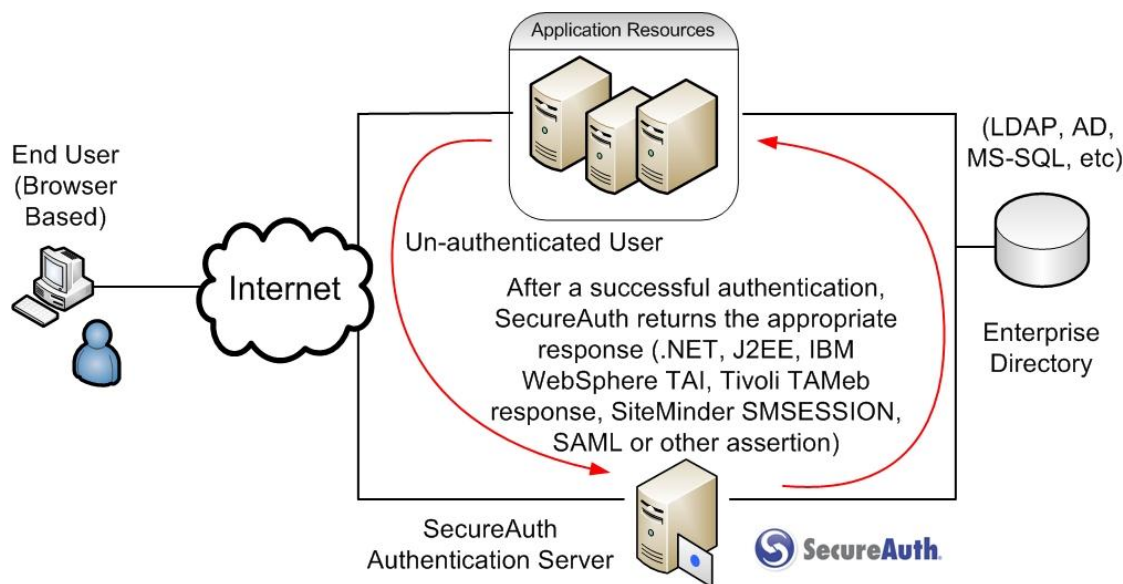


Figure #11 - The SecureAuth has the unique ability of integrated directly with the application requiring X.509 authentication without forcing the enterprise to “turn-on” C-SSL on the webserver.

4. PKI is Difficult for Application Integration – SecureAuth is NOT (cont'd)

SecureAuth integration to an existing application can be as simple as turning on Forms Based Authentication and directing the resource to the SecureAuth server. This is the case for applications such as Microsoft SharePoint/MOSS, Outlook Web Access and other ASP.NET applications. Similar target/redirect methodologies exist for J2EE applications and application servers such as IBM's WebSphere which utilizes a mechanism called Third-Party Lightweight Authentication (LPTA).

SecureAuth is currently integrated in the following environments:

- ASP.NET applications
- Microsoft Outlook Web Access
- Microsoft SharePoint/Moss applications
- J2EE applications
- CA SiteMinder
- Tivoli Access Manager for e-Business (TAMeb)
- Applications that can take a SAML (1.1 and 2.0) assertion
- Applications that accept OpenID authentication
- Any application that can read a GUID in share datastore
- SecureAuth has utilized this mechanism for applications in the "other" category

SecureAuth offers the most flexible authentication path, uniquely allowing an incremental approach to federation. SecureAuth becomes the authentication mechanism for the Identity Provider (IdP). Once implemented SecureAuth or another mechanism can create the SAML, OpenID, WS-* or other assertion that the Relying party is expecting. (See Figure #12)

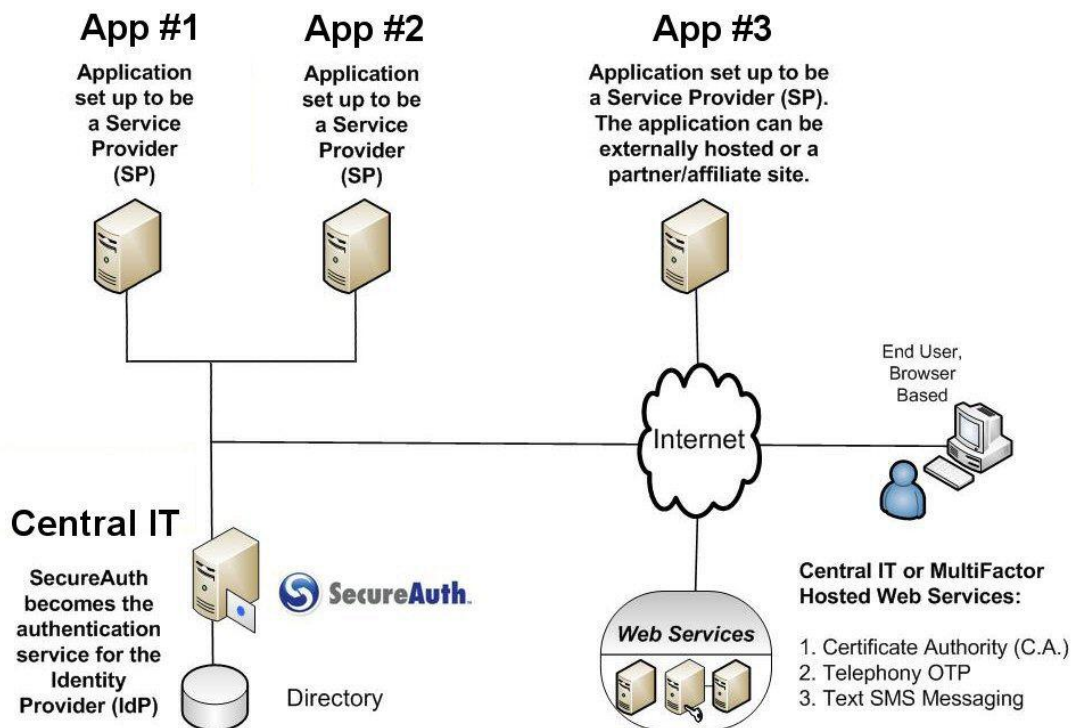


Figure #12 – SecureAuth can be the Authentication Mechanism for the Identity Provider in a federated environment.

5. PKI is Difficult for End Users to Obtain Credentials – SecureAuth is NOT

One of biggest “deal killers” of a PKI deployment is the difficulty associated in delivering the X.509 credentials to the end user. One central issue has been in educating end users on what mechanism to use to:

- Request a public/private key
- Store the public/private key
- Transport the public/private key

“Classic PKI” does not have an answer for this problem. End users simply do not have PKI backgrounds nor the understanding of the nuances and complexities of private and public keys. Attempts to put the credentials on portable key store have failed because of the lack of knowledge on the export and import process.

SecureAuth addresses all of these issues with:

- User Self-Registration (Figures #13-#18 below)
- One-Touch User Certificate Revocation (Figures #5, #6)

The user self-registration process is self-explanatory and requires no help desk support. Here is a step-thru of the process: (Figures #13-#18)

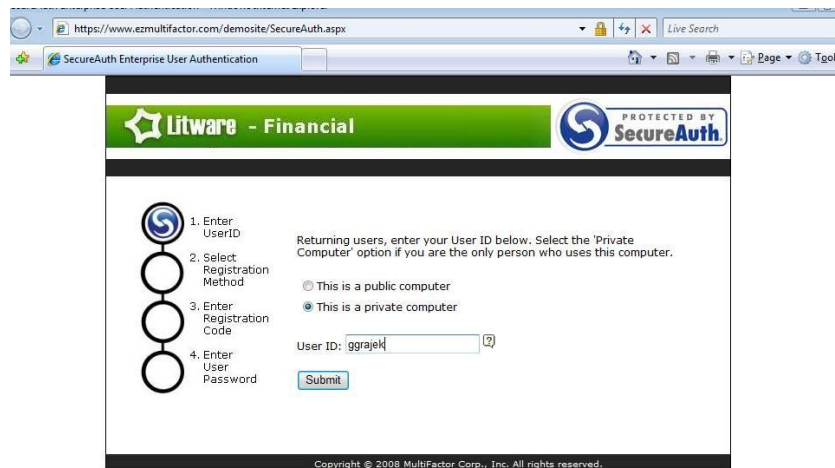


Figure #13: User enters his/her UserID to begin Self-Registration Process

5. PKI is Difficult for end users to obtain credentials – SecureAuth is NOT (cont'd)

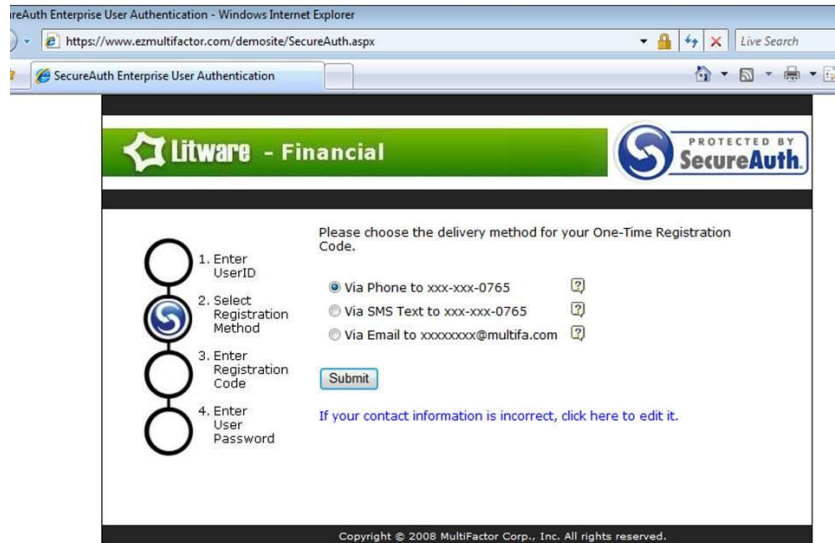


Figure #14: The user self-registers by selecting from a (enterprise-configurable) list of options

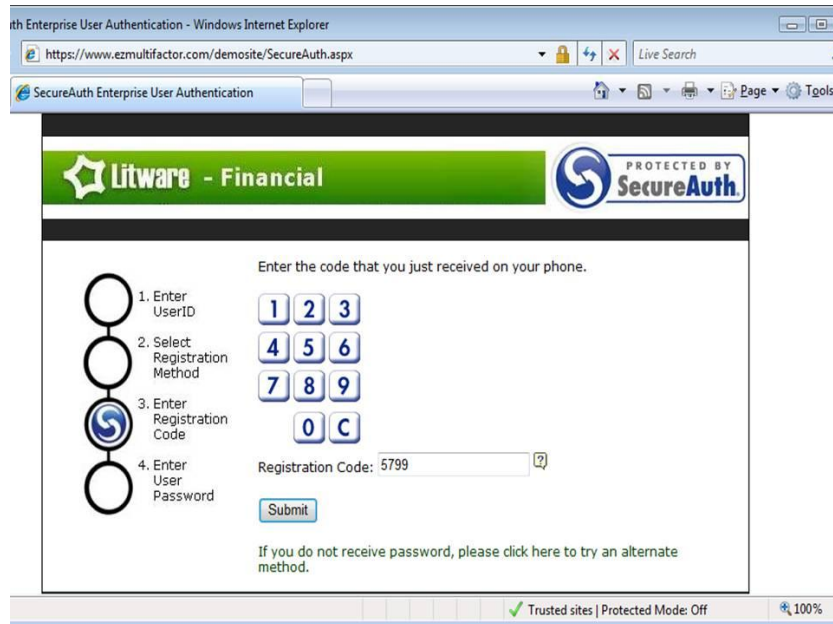


Figure #15: User enters One-Time-Registration Code via Java Keypad

5. PKI is Difficult for end users to obtain credentials – SecureAuth is NOT (cont'd)

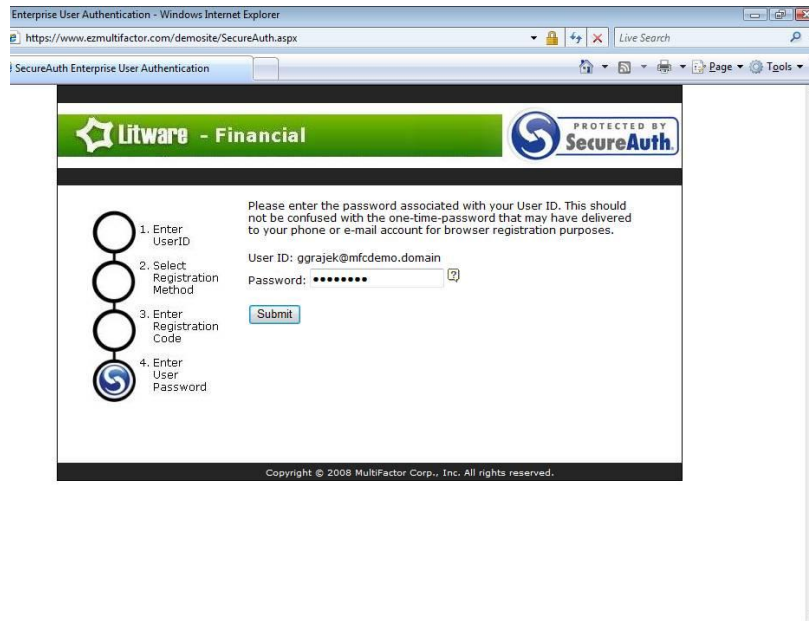


Figure #16: User inputs his enterprise (AD, LDAP MS-SQL, etc) password (Note: This is stored at the enterprise and not duplicated by SecureAuth)

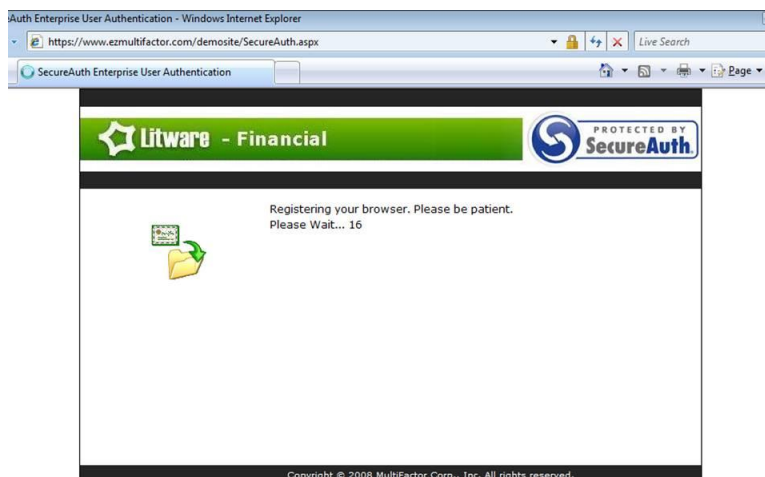
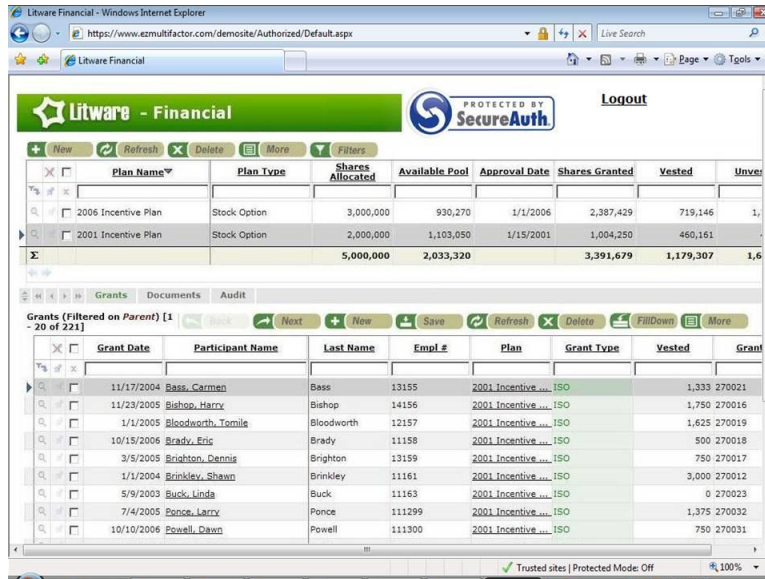


Figure #17: SecureAuth creates the public/private key pair on the end users device.

(The SecureAuth private key never transveres the network.)

5. PKI is Difficult for end users to obtain credentials – SecureAuth is NOT (cont'd)



Plan Name	Plan Type	Shares Allocated	Available Pool	Approval Date	Shares Granted	Vested	Unvested
2006 Incentive Plan	Stock Option	3,000,000	930,270	1/1/2006	2,387,429	719,146	1,668,283
2001 Incentive Plan	Stock Option	2,000,000	1,103,050	1/15/2001	1,004,250	460,161	543,889
Σ		5,000,000	2,033,320		3,391,679	1,179,307	1,668,283

Grant Date	Participant Name	Last Name	Empl #	Plan	Grant Type	Vested	Grant
11/17/2004	Bass, Carmen	Bass	13155	2001 Incentive...	ISO	1,333	270021
11/23/2005	Bishop, Harry	Bishop	14156	2001 Incentive...	ISO	1,750	270016
1/1/2005	Bloodworth, Tommie	Bloodworth	12157	2001 Incentive...	ISO	1,625	270019
10/15/2006	Brady, Eric	Brady	11158	2001 Incentive...	ISO	500	270018
3/5/2005	Brighton, Dennis	Brighton	13159	2001 Incentive...	ISO	750	270017
1/1/2004	Brinkley, Shawn	Brinkley	11161	2001 Incentive...	ISO	3,000	270012
5/9/2003	Buck, Linda	Buck	11163	2001 Incentive...	ISO	0	270023
7/4/2003	Ponce, Larry	Ponce	111299	2001 Incentive...	ISO	1,375	270032
10/10/2006	Powell, Dawn	Powell	111300	2001 Incentive...	ISO	750	270031

Figure #18: Lastly the user is redirected back to the Application

SecureAuth handles the creation of the public/private key for the end user. The self-explanatory, user self-registration process takes all the guesswork out of the registration process.

6. PKI is only Single Factor Authentication – SecureAuth is true 2-Factor

One of the biggest security concerns that eliminate PKI from authentication decisions is that PKI is actually only single factor authentication. The argument is that the choices for “factors” are:

- Something you have
- Something you know

PKI, the authentication factor is only the first, something you have. And the security argument is: “if the device is compromised or stolen” – there is no other factor since the certificate is the only factor.

It’s important to note here, that there are new “PKI” solutions on the market being offered. These solutions promise to add a factor into the equation by utilizing a password in the “unlocking” of the key store. Security analysts have accurately stated that these solutions are not true 2-factor authentication. The fact that the password is locked with the credential simply means the equation is purely 1-factor and the attack method is simply a “brute force” attack on the credential to “guess” the password. In this manner, the security analysts are correct – these solutions are simply 2-factor since the certificate can be removed from the device and then brute force attacked. Once the certificate is unlocked from these means, the credential can be replayed and the identity of the legitimate user impersonated by the hacker.

SecureAuth is unique in the X.509 authentication market by its ability to conduct its own validation of the X.509 credential. That is SecureAuth is able to conduct the public/private key pair exchange that occurs in the SecureAuth authentication dialogue. SecureAuth does not rely on legacy PKI code to in validation – instead it has the application redirect the authentication to the SecureAuth authentication appliance for the authentication dialogue. (See Figure #19)

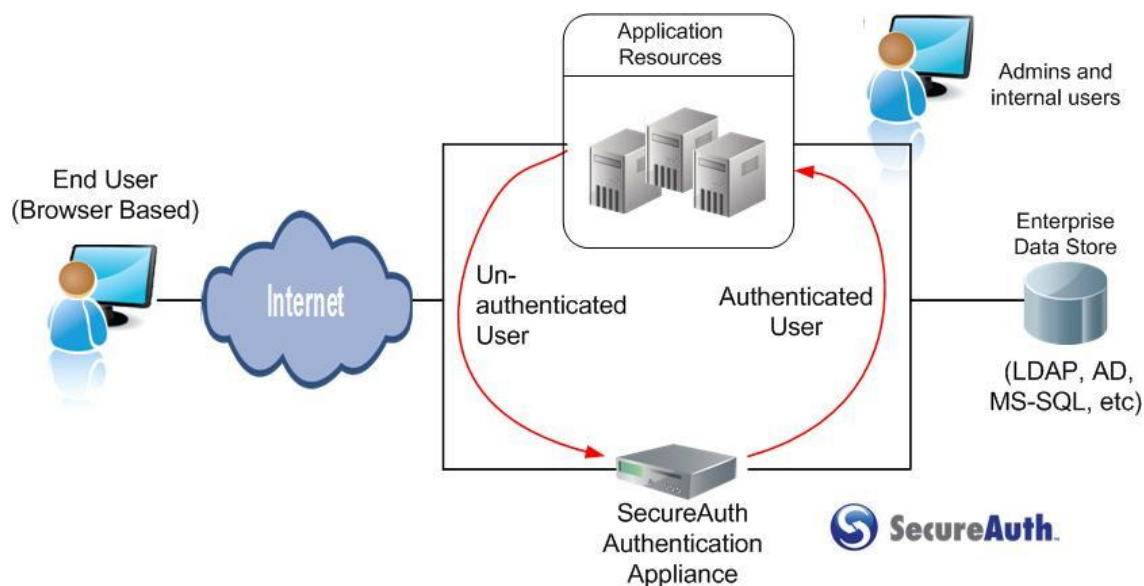


Figure #19 – SecureAuth conducts its own client/server validation

6. PKI is Single Factor Authentication – SecureAuth is true 2-Factor (Cont'd)

Because SecureAuth is not relying on legacy, circa 1988 validation code, SecureAuth is able to conduct a true 2-Factor, 2-way authentication, utilizing the original and valid public and private key infrastructure.

The SecureAuth 1st Factor - X.509 Authentication

In a SecureAuth X.509 validation, the user's X.509v3 credential (the first) factor is validated via the following means:

1. User's public Key Sent to SecureAuth Server
2. Certificate Request Identifier (CRI), a SecureAuth authentication nonce
3. The SecureAuth X.509 server information SSL Certificate
4. The SecureAuth Authentication URL

This information packet is then signed with the user's private key, by the SecureAuth client and then returned to the SecureAuth server on validation. SecureAuth is insuring the legitimacy of the user by implementing a public/private key pair signing of the user's SecureAuth's X.509 key pair. In addition, SecureAuth is validating that the end user is dialoguing with the legitimate SecureAuth server in (2) ways:

1. The SecureAuth client, via a 2nd SSL request obtains and signs, with user's private key the SecureAuth X.509 identity certificate
2. The SecureAuth server signs the SecureAuth CRI (Certificate Request Identifier) with the SecureAuth private key and then encrypts with the SecureAuth user's public key

This 2-step mechanism to insure that the end user is communicating with the legitimate server, is a valid "defense in depth" procedure. The first mechanism ensures that the SSL Termination point for the SecureAuth server is the legitimate destination. The 2nd mechanism ensures that the SecureAuth authentication packet is not tampered with inside the "corporate cloud" – e.g. the SSL termination point may exist away from the SecureAuth server. For this and for other security reasons, SecureAuth has a 2nd defense layer. SecureAuth signs the authentication once with the SecureAuth server private and then enacts an encryption with the user's public key. This mechanism insures that only the SecureAuth server is able to validate the SecureAuth authentication packet – and not another illegitimate authentication server in the enterprise's corporate cloud.

It is important to note, the SecureAuth X.509 credential is not encrypted with the user's password. As stated previously – this method simply makes the credential vulnerable to brute-force decryption attacks. Instead, the SecureAuth credential is "fingerprinted" with the information germane to that particular device. In this manner, the credential is NOT valid if removed and is not vulnerable to the off-device brute force attack method. If the end-user desires portability, the end-user simply steps through the SecureAuth unique user self-enrollment functionality, (see pages 13-16).

6. PKI is Single Factor Authentication – SecureAuth is True 2-Factor (Cont'd)

The SecureAuth 2nd Factor – Encrypted credentials from the Enterprise Data Store

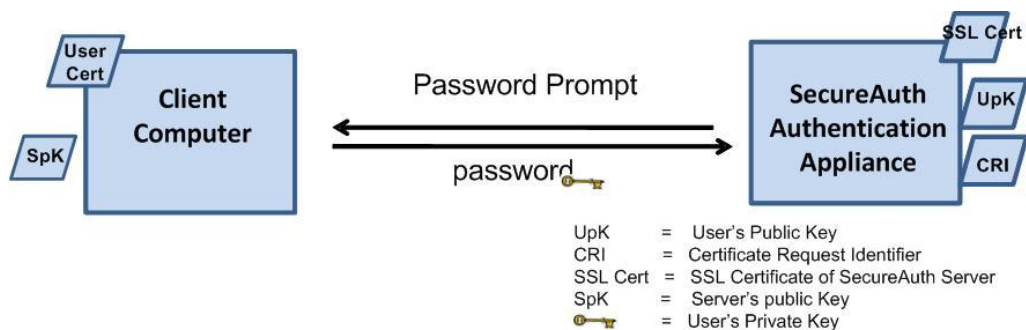
The above scenario details how the SecureAuth X.509 key exchange thwarts man-in-the-middle and other replay attacks with a true “Something you have” authentication credential. This factor is the SecureAuth X.509v3 key pair. As detailed, SecureAuth does not tamper with the credential with the user’s password and claim this to be a 2nd factor – as Arcot and Tricipher do – instead SecureAuth fingerprints the credential with information that can be only be valid from authentication device.

Once SecureAuth has validated the X.509 key exchange, as detailed above, the user is asked to submit a credential that is NOT stored on the SecureAuth appliance – but is actually obtained from the enterprise’s data store. Because SecureAuth is conducting both the left (client) and right (server) side dialogue in this discussion, SecureAuth is able to ensure that this credential is not inspected, stored and replayed in an attack. (See **Figure #20**)

In the SecureAuth 2nd Factor scenario, the SecureAuth prompts the end user for his password. This is NOT a SecureAuth retained credential – but then enterprise retained password. (See **Figure #19**).

SecureAuth does not just rely on the SSL connection for a safe transmission of this factor. Instead it utilizes the SecureAuth components that made the 1st factor, the X.509 certificate authentication valid. The SecureAuth JRE client, upon receipt of the user’s password, signs the credential with the user’s private key and then encrypts the password with the SecureAuth server’s public key. Once again, this insures that:

- The SecureAuth server can be sure the password came from the client’s machine
- The User can be assured that only the SecureAuth server can decrypt and utilize the password.



The SecureAuth client, on submit of the password:

- a) Signs the password with the Server’s public key
- b) Encrypts the password with the server’s public key

And then transmit credential packet to SecureAuth server.

Figure #20 - SecureAuth ensures that the User’s Credential, the 2nd Factor, is not replayed or altered in transmission



6. PKI is Single Factor Authentication – SecureAuth is True 2-Factor (Cont'd)

Summary:

Because the password, the SecureAuth 2nd Factor, is obtained from the enterprise's data store and is NOT tied to credential, SecureAuth can legitimately be called a true 2-factor authentication product. It is important to note that SecureAuth uniquely, encrypts the user's password separately and signs it to insure confidentiality and to insure that man-in-the-middle attack is occurring.