(12) INTERNATIONAL APPLICATION PUBLISHED UNDER THE PATENT COOPERATION TREATY (PCT)

(19) World Intellectual Property Organization
International Bureau

(43) International Publication Date
15 September 2011 (15.09.2011)

PCT

(10) International Publication Number
WO 2011/112466 A2

(51) International Patent Classification:
H04L 9/32 (2006.01) *G06F 21/20* (2006.01)

(21) International Application Number:
PCT/US2011/027297

(22) International Filing Date:
4 March 2011 (04.03.2011)

(25) Filing Language: English

(26) Publication Language: English

(30) Priority Data:
12/719,853 8 March 2010 (08.03.2010) US

(71) Applicant *(for all designated States except US)*: **MICROSOFT CORPORATION** [US/US]; One Microsoft Way, Redmond, WA 98052-6399 (US).

(72) Inventors: **SHARIF, Tariq**; c/o Microsoft Corporation, LCA - International Patents, One Microsoft Way, Redmond, WA 98052-6399 (US). **BRACE, Colin, H.**; c/o Microsoft Corporation, LCA - International Patents, One Microsoft Way, Redmond, WA 98052-6399 (US). **GARG, Nakul**; c/o Microsoft Corporation, LCA - International Patents, One Microsoft Way, Redmond, WA 98052-6399 (US).

(81) Designated States *(unless otherwise indicated, for every kind of national protection available)*: AE, AG, AL, AM, AO, AT, AU, AZ, BA, BB, BG, BH, BR, BW, BY, BZ, CA, CH, CL, CN, CO, CR, CU, CZ, DE, DK, DM, DO, DZ, EC, EE, EG, ES, FI, GB, GD, GE, GH, GM, GT, HN, HR, HU, ID, IL, IN, IS, JP, KE, KG, KM, KN, KP, KR, KZ, LA, LC, LK, LR, LS, LT, LU, LY, MA, MD, ME, MG, MK, MN, MW, MX, MY, MZ, NA, NG, NI, NO, NZ, OM, PE, PG, PH, PL, PT, RO, RS, RU, SC, SD, SE, SG, SK, SL, SM, ST, SV, SY, TH, TJ, TM, TN, TR, TT, TZ, UA, UG, US, UZ, VC, VN, ZA, ZM, ZW.

(84) Designated States *(unless otherwise indicated, for every kind of regional protection available)*: ARIPO (BW, GH, GM, KE, LR, LS, MW, MZ, NA, SD, SL, SZ, TZ, UG, ZM, ZW), Eurasian (AM, AZ, BY, KG, KZ, MD, RU, TJ, TM), European (AL, AT, BE, BG, CH, CY, CZ, DE, DK, EE, ES, FI, FR, GB, GR, HR, HU, IE, IS, IT, LT, LU, LV, MC, MK, MT, NL, NO, PL, PT, RO, RS, SE, SI, SK, SM, TR), OAPI (BF, BJ, CF, CG, CI, CM, GA, GN, GQ, GW, ML, MR, NE, SN, TD, TG).

Declarations under Rule 4.17:

— *as to applicant's entitlement to apply for and be granted a patent (Rule 4.17(ii))*

— *as to the applicant's entitlement to claim the priority of the earlier application (Rule 4.17(iii))*

[Continued on next page]

(54) Title: AUTOMATED CERTIFICATE MANAGEMENT

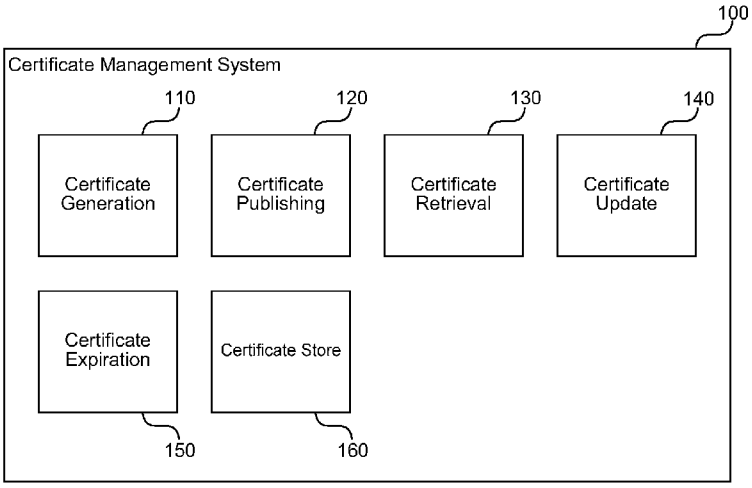


FIG. 1

(57) Abstract: A certificate management system provides automated management of certificate lifecycles and certificate distribution. Rather than depend upon an administrator to manually distribute and manage certificates, the system self-generates certificates, distributes the certificates to appropriate servers or other parties, and transitions from old certificates to new certificates in a well-defined manner that avoids breaking functionality. After generating one or more certificates, the system securely shares certificates in a way that parties that use them can find the new certificates without an administrator manually distributing the certificates. When it is time to update certificates, the system generates new certificates and shares the new certificates in a similar way. During a transition period, the system provides a protocol by which both old and new certificates can be used to perform authenticated access to resources, so that the transition from an old to a new certificate does not break services.

WO 2011/112466 A2

Published:

— *without international search report and to be republished upon receipt of that report (Rule 48.2(g))*

AUTOMATED CERTIFICATE MANAGEMENT

BACKGROUND

[0001] Many organizations and systems include server computer systems that provide various important functions to the organization. For example, even a small business may have an email server, one or more database servers, a company intranet portal hosted by a web server, a file server, and so forth. These functions may reside on a single computer system or on several computer systems, such as to increase scalability and provide adequate resources for each function based on anticipated or measured loads. Products, such as MICROSOFT TM WINDOWS TM Small Business Server provide multiple server functions in a single product.

[0002] When an organization has multiple servers, the servers may include various authentication mechanisms and security protocols to allow servers to trust one another and to allow users to sign on to access the functions provided by each server. One security system for allowing servers to communicate using a single sign-on is MICROSOFT TM ACTIVE DIRECTORY TM Federation Services (ADFS). ADFS was introduced in MICROSOFT TM WINDOWS SERVER TM 2003 for organizations to participate in standards-based identity federation. ADFS provides claims-based authentication across servers and even across organizations. Claims based authentication uses tokens and certificates to authenticate a user or server with a set of claims contained within the token.

[0003] Unfortunately, introducing certificates also introduces the problem of certificate management, as certificates expire over time. In a product deployment that uses Public Key Infrastructure (PKI) or similar policies to perform some of its functions, it is very common to assume that the product relies on an administrator to manually manage the certificate lifecycle. In addition, such products rely on the administrator to do the heavy lifting of sharing these certificates in a server farm. For many organizations, particularly small businesses, where sophisticated administrators are typically not employed, this assumption often leads to loss of functionality and servers that cannot communicate correctly to provide the expected functions to the organization. Even when an administrator does replace an old certificate with a new one as scheduled, there is a chance that something in a system might break because of this sudden change, by not giving a warning to others dependent on the system about this change ahead of time. In addition, certificate updates involve significant planning as certificates are typically updated manually on each server on the same day as part of scheduled downtime.

SUMMARY

[0004] A certificate management system is described herein that provides automated management of certificate lifecycles and certificate distribution. Rather than depend upon an administrator or other user to manually distribute and manage certificates, the certificate management system self-generates certificates, distributes the certificates to appropriate servers or other parties, and transitions from old certificates to new certificates in a well-defined manner that avoids breaking functionality. After generating one or more certificates, the system securely shares certificates in a way that parties that use them can find the new certificates without an administrator manually distributing the certificates. When it is time to update certificates, the system generates new certificates and shares the new certificates in a similar way. During a transition period, the system provides a protocol by which both old and new certificates can be used to perform authenticated access to resources, so that the transition from an old to a new certificate does not break services. Sometime later, the system removes the old certificates and the cycle repeats as the new certificates approach expiration. During each of these phases, no administrative intervention is expected or required. Thus, the certificate management system greatly simplifies the process of establishing trust between the various servers of an organization and relieves users from diagnosing complex problems that commonly occur under manual management of certificate lifetimes.

[0005] This Summary is provided to introduce a selection of concepts in a simplified form that are further described below in the Detailed Description. This Summary is not intended to identify key features or essential features of the claimed subject matter, nor is it intended to be used to limit the scope of the claimed subject matter.

BRIEF DESCRIPTION OF THE DRAWINGS

[0006] Figure 1 is a block diagram that illustrates components of the certificate management system, in one embodiment.

[0007] Figure 2 is a flow diagram that illustrates processing of the system to generate and share certificates, in one embodiment.

[0008] Figure 3 is a flow diagram that illustrates processing of the system to update one or more certificates, in one embodiment.

DETAILED DESCRIPTION

[0009] A certificate management system is described herein that provides automated management of certificate lifecycles and certificate distribution. Rather than depend upon an administrator or other user to manually distribute and manage certificates, the

certificate management system self-generates certificates, distributes the certificates to appropriate servers or other parties, and transitions from old certificates to new certificates in a well-defined manner that avoids breaking functionality. After generating one or more certificates, the system securely shares certificates in a way that parties that use them can find the new certificates without an administrator manually distributing the certificates. For example, a certificate server in a small business may provide certificates to other servers (e.g., an email server) without manual intervention. When it is time to update certificates, the system generates new certificates and shares the new certificates in a similar way. For example, some time prior to certificate expiration, the system may create replacement certificates for soon-to-expire certificates. During a transition period, the system provides a protocol by which both old and new certificates can be used to perform authenticated access to resources, so that the transition from an old to a new certificate does not break services. Sometime later, the system removes the old certificates and the cycle repeats as the new certificates approach expiration. During each of these phases, no administrative intervention is expected or required. Thus, the certificate management system greatly simplifies the process of establishing trust between the various servers of an organization and relieves users from diagnosing complex problems that commonly occur under manual management of certificate lifetimes.

[0010] In some embodiments, upon first configuring a new server farm using the certificate management system, the system creates its own self-signed/issued certificates for token signing and encryption. The system publishes these certificates to other security token services or applications as federation metadata using a standard protocol. If the system is configured for farm configuration, the system encrypts these certificates with a symmetric key and shares the encrypted certificates with other certificate management machines in the farm. The symmetric key (or a key derivation key from which others servers can derive the symmetric key) is also saved and made accessible to other machines in the farm using a secure method. After the system creates certificates, the system starts monitoring the certificates regularly (e.g., daily) to determine whether the certificates are about to expire. A predetermined time before the expiration of the certificates (e.g., a couple of weeks), the system creates new certificates and starts to publish the new certificates as secondary certificates (e.g., using the standard called federation metadata). Publishing these certificates as secondary certificates ensures that no security token service or application is broken with a quick expiration of old certificates and gives other parties enough time to make appropriate changes on their side. After creating the new

certificates, the certificate management system automatically shares the new certificates with other machines in the farm using the symmetric key and does not rely on an administrator to share certificates. Once the originally created certificates expire, the system deletes them. At a configurable time after creating new certificates and before the old certificates expire, the system promotes the new certificates to primary certificates. The time between generating/publishing new certificates and promoting the new certificates to primary defines the transition period. The system then repeats the process by monitoring the new certificates and managing their life cycle.

[0011] Figure 1 is a block diagram that illustrates components of the certificate management system, in one embodiment. The system 100 includes a certificate generation component 110, a certificate publishing component 120, a certificate retrieval component 130, a certificate update component 140, a certificate expiration component 150, and a certificate store 160. Each of these components is described in further detail herein.

[0012] The certificate generation component 110 generates one or more certificates for token signing and encryption. The component 110 creates an initial set of certificates upon installation of the system 100 and configuration of a server farm for automated certificate management. Later, as the initial set of certificates is close to expiration, the certificate generation component 110 creates new certificates to replace the old ones. The component 110 repeats this process over time to maintain a current set of certificates that is available to perform encryption and signing services needed by application servers.

[0013] The certificate publishing component 120 publishes new certificates created by the certificate generation component 110 to make the certificates available to other servers with which the system 100 communicates. The certificate publishing component 120 may use standard protocols such as federation metadata and WS-Trust to make new certificates available upon request to other servers. A primary server may exchange a symmetric key in a secure manner with other servers and encrypt certificates using the symmetric key. In this way, certificates can be shared in the open (e.g., via a file share) but can only be accessed and used by those with an appropriate level of access (as demonstrated by possession of the symmetric key). When publishing to members of the same farm, the component 120 publishes the certificate in an encrypted manner and the content includes the public as well as the private key of the certificate. In some embodiments, this publishing is done via a web service internal to the farm (e.g., using WINDOWS TM Communication Foundation (WCF). When publishing to external trust partners of the

ADFS farm, the component 120 may just publish the public key of the certificates and this publishing is done using externally visible federation metadata. In some embodiments, the component 120 may include separate subcomponents for publishing to external parties and for publishing to internal farm members.

[0014] The certificate publishing component 120 may publish both primary and secondary certificates. For example, the initially created set of certificates are primary certificates, but later replacement certificates may initially be published as secondary certificates for some period of time to allow a smooth transition between old and new certificates. After the transition period, the certificate publishing component 120 publishes the replacement certificates as primary certificates.

[0015] The certificate retrieval component 130 operates on a server that depends on the certificate management system 100 to retrieve new certificates shared by the certificate publishing component 120. Application servers or other servers may periodically poll a primary server of the system to determine whether new certificates are available for download to the application server's certificate store.

[0016] The certificate update component 140 operates on a server that depends on the certificate management system 100 to add primary certificates retrieved from the system 100 through the certificate publishing component 120 to a certificate store local to a server in the server farm. The certificate update component 140 also retrieves secondary certificates and places the secondary certificates into a certificate store in a manner that accesses can be made by clients using either the primary or secondary certificates during a transition period. Thus, even if a particular server has failed to update to a set of replacement certificates, clients can still communicate with the particular server using one of the primary or secondary certificates. By the time the system 100 promotes the secondary certificates to primary certificates (e.g., at the close of the transition period), all servers will have had sufficient time to update to the latest set of certificates.

[0017] The certificate expiration component 150 removes expired certificates from the system 100 and promotes secondary replacement certificates to primary certificates. Once a certificate is beyond its lifetime, servers will no longer be using the certificate for encryption and signing, so the component 150 deletes certificates that have expired. By this time, the system 100 has distributed a set of secondary certificates as replacements as described herein, and the component 150 promotes the secondary certificates to primary certificates.

[0018] The certificate store 160 stores certificates for ADFS generated by the certificate generation component 110. The certificate store 160 may include primary certificates and secondary certificates. In addition, individual application servers may include their own certificate store 160 for storing downloaded primary and secondary certificates for encrypting and signing data. There may be a separate cryptographic store provided by a cryptographic service provider (CSP) that is not shown.

[0019] Although the above components are described above in the context of a single certificate management system 100, those of ordinary skill in the art will recognize that the components can be distributed on various servers and computer systems to carry out the functions described herein. For example, each server in an organization may include its own certificate expiration component 150 that removes old and promotes new certificates for that server. Alternatively or additionally, a primary certificate server may remove old certificates on one or more servers in a centrally managed manner. The system is not limited to any particular distribution of components and is compatible with various distributions that fulfill the purposes and functions described herein.

[0020] The computing device on which the certificate management system is implemented may include a central processing unit, memory, input devices (e.g., keyboard and pointing devices), output devices (e.g., display devices), and storage devices (e.g., disk drives or other non-volatile storage media). The memory and storage devices are computer-readable storage media that may be encoded with computer-executable instructions (e.g., software) that implement or enable the system. In addition, the data structures and message structures may be stored or transmitted via a data transmission medium, such as a signal on a communication link. Various communication links may be used, such as the Internet, a local area network, a wide area network, a point-to-point dial-up connection, a cell phone network, and so on.

[0021] Embodiments of the system may be implemented in various operating environments that include personal computers, server computers, handheld or laptop devices, multiprocessor systems, microprocessor-based systems, programmable consumer electronics, digital cameras, network PCs, minicomputers, mainframe computers, distributed computing environments that include any of the above systems or devices, and so on. The computer systems may be cell phones, personal digital assistants, smart phones, personal computers, programmable consumer electronics, digital cameras, and so on.

[0022] The system may be described in the general context of computer-executable instructions, such as program modules, executed by one or more computers or other devices. Generally, program modules include routines, programs, objects, components, data structures, and so on that perform particular tasks or implement particular abstract data types. Typically, the functionality of the program modules may be combined or distributed as desired in various embodiments.

[0023] Figure 2 is a flow diagram that illustrates processing of the system to generate and share certificates, in one embodiment. Beginning in block 210, the system generates one or more certificates for token signing and encryption. The certificates are self-signed/self-issued. The system may create certificates of a variety of purposes. For example, the system may create a general certificate used by servers in a server farm to establish trust with one another, a certificate for a remote server to use to communicate with the server farm, and so forth. Continuing in block 220, the system encrypts the certificates using a key securely shared among servers in a server farm. For example, the system may generate and use a symmetric key to encrypt the one or more generated certificates. Encrypting the certificates allows the certificates to be shared more freely without worrying that unauthorized parties can access the certificates.

[0024] Continuing in block 230, the system publishes the certificates at a location available to each server in the server farm. For example, the system may share the certificates at a file share or via a common protocol at a port of a certificate server. The system may also indicate a purpose for each certificate and whether the certificate is a primary or secondary certificate. Sometime later, one or more servers request published certificates as described by the following steps. Continuing in block 240, the system listens for requests to access published certificates. For example, the system may listen at a well-known port for server requests to access certificates associated with the server farm. Continuing in decision block 250, if the system receives a request to access a certificate, then the system continues at block 260, else the system loops to block 240 to continue listening for requests.

[0025] Continuing in block 260, the system processes the received request to access a certificate. For example, a server in a server farm may periodically poll the certificate server for new certificates. Continuing in block 270, the system replies with the requested certificate. In some embodiments, servers may request multiple certificates or certificates newer than a specified date/time from the certificate server, and the system replies with

one or more certificates that meet the specified criteria. After block 270, the system loops to block 240 to listen for additional certificate access requests.

[0026] Figure 3 is a flow diagram that illustrates processing of the system to update one or more certificates, in one embodiment. Beginning in block 310, the system detects pending expiration of a certificate. For example, the system may periodically poll certificates stored in a certificate store to identify certificates close to expiration (e.g., within a week or two of expiration). The system may receive a configurable parameter that specifies a duration before expiration to start updating certificates. The system may handle the expiration of many certificates at the same time although a single certificate is described in this example. Continuing in block 320, the system generates a new certificate to replace the expiring certificate. For example, the system may invoke an operating system application programming interface (API) or other component for generating certificates.

[0027] Continuing in block 330, the system identifies the new certificate as a secondary certificate, so that servers retrieving the certificate will continue to use existing certificates as well as accepting requests that use the new certificate. Continuing in block 340, the system publishes the new certificate at a location available to each server in a server farm. Servers will pick up the new certificate automatically and begin allowing accesses that rely on the new certificate. Publishing the new certificate may also include publishing the certificate to external partners outside of the ADFS farm. Continuing in block 350, the system detects whether a transition period has ended, and if so continues at block 360, else the system loops to block 350 to wait for the end of the transition period. The transition period ends when the certificate expires or sometime after certificate expiration (it may be useful to allow a certificate to stay valid until slightly after expiration).

[0028] Continuing in block 360, the system promotes the new certificate from a secondary certificate to a primary certificate. This completes the replacement of the expiring certificate with the new certificate. The system may simply mark the certificate differently in the certificate store so that dependent servers notice the change and update their own certificate stores to indicate the new primary certificate. Continuing in block 370, the system removes the expired certificate from the certificate store. This may also prompt dependent servers to remove the expired certificate if they have not already removed it according to their own expiration management schedules. The transition period provided ensures that no security token service or application is broken with a quick expiration of old certificates and gives other parties enough time to make appropriate changes on their side. After block 370, these steps conclude.

[0029] In some embodiments, the certificate management system provides each server in a server farm with a common service account through which each server can access shared certificates. For example, by sharing certificates in a store that is marked as accessible by the service account, any other server that holds the credentials of the service account can access the store and any data within the store. This provides an easy way of sharing data among multiple machines in a secure manner that is already provided by existing server user access management.

[0030] In some embodiments, the certificate management system provides configurable parameters that an administrator can adjust to increase security or to increase the likelihood of a successful transition between certificates. For example, some administrators may want a shorter transition period for added security, while others may prefer a longer transition period for added compatibility. The system allows an administrator to configure the transition period to facilitate tuning the system to accommodate such differing goals.

[0031] In some embodiments, the certificate management system publishes metadata that indicates information such as which certificates are in use, which are expired, which are primary, which are secondary, and so forth. In this way, servers in a server farm can periodically check the published metadata to determine if it is time for further action, such as updating or promoting certificates.

[0032] In some embodiments, the certificate management system retrieves certificates from a separate certificate authority (CA). Although described herein as generating its own certificates, the system can also defer to a particular CA trusted by accessing servers to generate certificates shared and managed by the system. For example, the system may invoke the WINDOWS SERVER TM 2008 R2 web services API to get certificates from a public CA.

[0033] From the foregoing, it will be appreciated that specific embodiments of the certificate management system have been described herein for purposes of illustration, but that various modifications may be made without deviating from the spirit and scope of the invention. Accordingly, the invention is not limited except as by the appended claims.

CLAIMS

1. A computer-implemented method for automatically generating and sharing security certificates, the method comprising:

generating one or more certificates for token signing and encryption in a server farm;

encrypting the certificates using a key securely shared among servers in the server farm;

publishing the certificates at a location available to each server in the server farm;

listening for requests to access published certificates; and

upon receiving a request to access a certificate ,

processing the received request to access a certificate; and

replying to the processed request with the requested certificate,

wherein the preceding steps are performed by at least one processor.

2. The method of claim 1 wherein generating one or more certificates comprises invoking a local application programming interface (API) for creating certificates.

3. The method of claim 1 wherein generating one or more certificates comprises invoking a remote public certificate authority (CA) to generate certificates.

4. The method of claim 1 wherein encrypting the certificates comprises accessing the key using a common service account used by each server in the server farm to access shared certificate information.

5. The method of claim 1 wherein encrypting the certificates comprises generating and storing a symmetric key or a key derived from the symmetric key to encrypt the one or more generated certificates.

6. The method of claim 1 wherein publishing the certificates comprises placing the certificates in a public location in the server farm secured using a common service account so that only authorized parties can access the published certificates.

7. The method of claim 1 wherein publishing the certificates comprises identifying each certificate as a primary or secondary certificate.

8. The method of claim 1 wherein listening for requests to access published certificates comprises listen at a well-known port for server requests to access certificates associated with the server farm.

9. The method of claim 1 wherein processing the received request comprises identifying the request as a periodic poll from a server in the server farm to detect newly generated or promoted certificates.

10. The method of claim 1 wherein processing the received request comprises identifying certificates that have changed in a certificate collection and wherein replying to the processed request comprises sending the identified certificates in response to the request.

11. A computer system for automatically managing certificates shared by multiple servers, the system comprising:

a processor and memory configured to execute software instructions;

a certificate generation component configured to generate one or more certificates for token signing and encryption;

a certificate publishing component configured to publish new certificates created by the certificate generation component to make the certificates available to other servers with which the system communicates;

a certificate retrieval component configured to retrieve new certificates shared by the certificate publishing component;

a certificate update component configured to add primary certificates retrieved from the certificate publishing component to a certificate store local to a server in the server farm;

a certificate expiration component configured to remove expired certificates from the system and promotes secondary replacement certificates to primary certificates; and

a certificate store configured to store certificates generated by the certificate generation component.

12. The system of claim 11 wherein the certificate generation component is further configured to create an initial set of certificates upon installation of the system and configuration of a server farm for automated certificate management.

13. The system of claim 11 wherein the certificate generation component is further configured to create new certificates to replace old ones nearing expiration without intervention by an administrator.

14. The system of claim 11 wherein the certificate publishing component is further configured to publish both primary and secondary certificates to allow a transition period between original and replacement certificates as certificates approach expiration.

15. The system of claim 11 wherein the certificate update component is further configured to retrieve secondary certificates and place the secondary certificates into a certificate store in a manner that accesses can be made to other servers using either the primary or secondary certificates during a transition period.

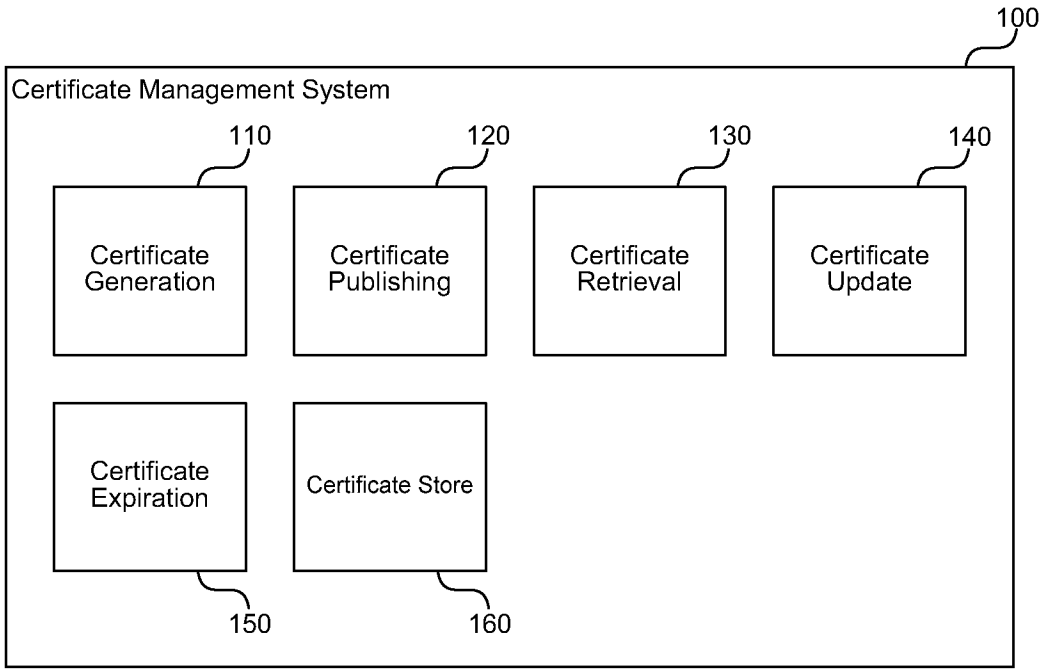

100
Certificate Management System
110
120
130
140
Certificate Generation
Certificate Publishing
Certificate Retrieval
Certificate Update
Certificate Expiration
Certificate Store
150
160

FIG. 1

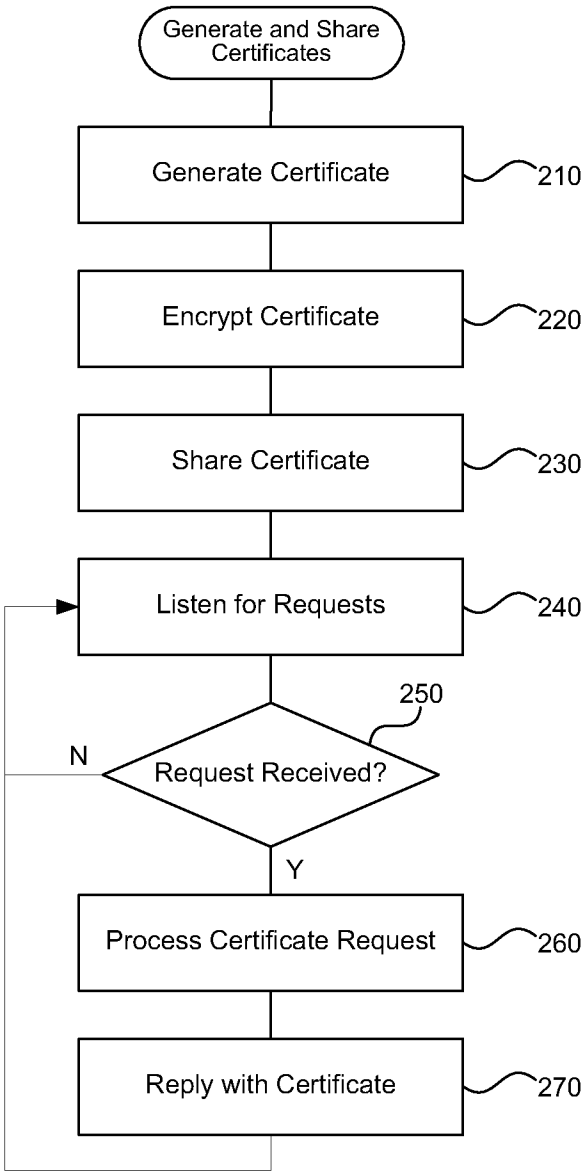

Generate and Share Certificates
Generate Certificate
210
Encrypt Certificate
220
Share Certificate
230
Listen for Requests
240
250
Request Received?
N
Y
Process Certificate Request
260
Reply with Certificate
270

FIG. 2

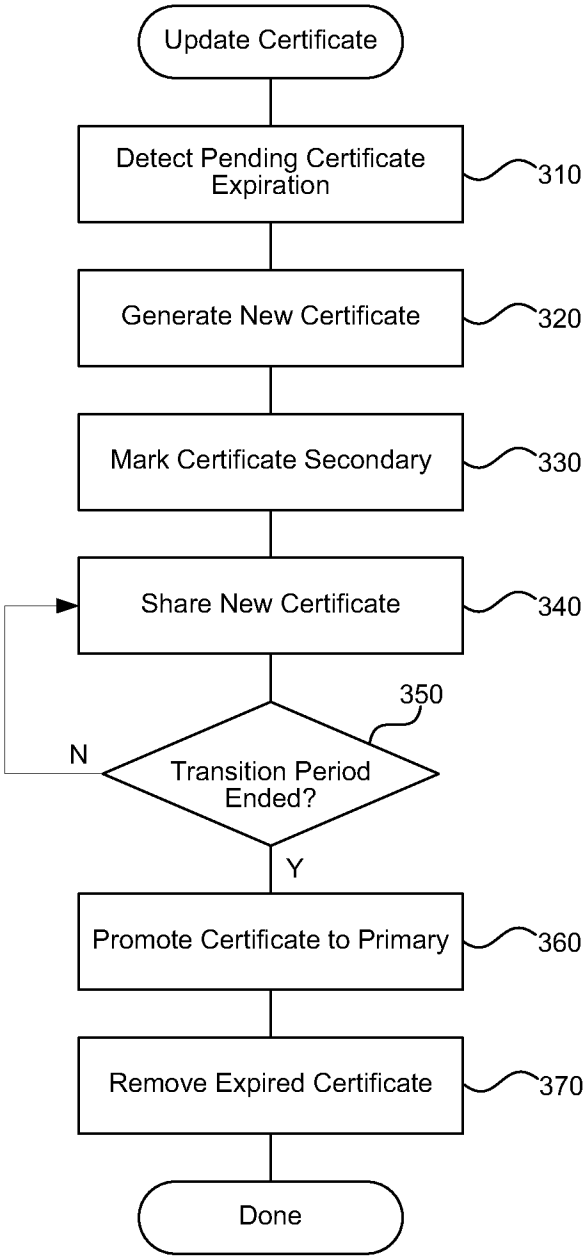

Update Certificate
Detect Pending Certificate Expiration
310
Generate New Certificate
320
Mark Certificate Secondary
330
Share New Certificate
340
350
Transition Period Ended?
N
Y
Promote Certificate to Primary
360
Remove Expired Certificate
370
Done

FIG. 3