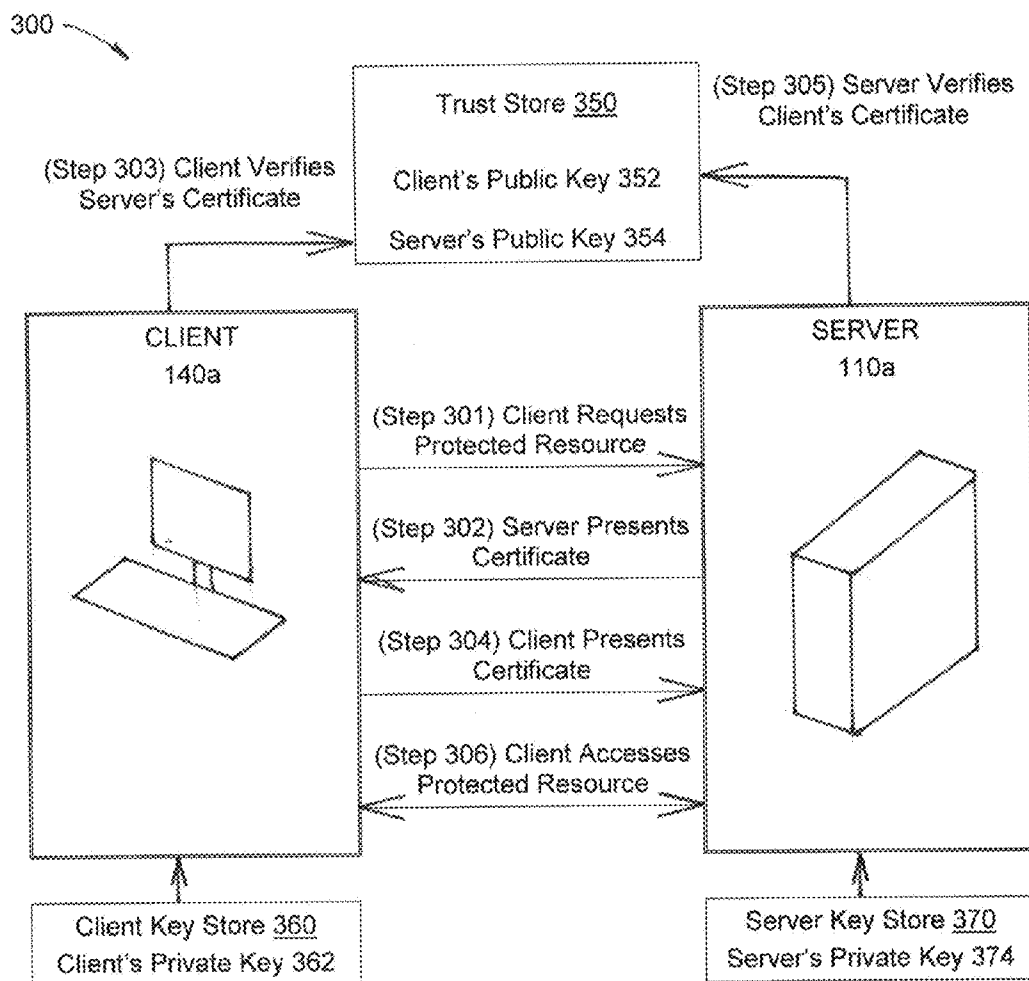




US 20120324225A1

(19) **United States**(12) **Patent Application Publication**
Chambers et al.(10) **Pub. No.: US 2012/0324225 A1**(43) **Pub. Date: Dec. 20, 2012**(54) **CERTIFICATE-BASED MUTUAL
AUTHENTICATION FOR DATA SECURITY****Publication Classification**(51) **Int. Cl.**
H04L 9/28 (2006.01)
(52) **U.S. Cl.** **713/169**(76) Inventors: **Jason Chambers**, Dunwoody, GA (US);
Theresa Robison, Sandy Springs, GA
(US); **Dameion Dorsner**, Atlanta, GA
(US); **Sridhar Manickam**, Alpharetta,
GA (US); **Daniel Konisky**, Atlanta, GA
(US)(57) **ABSTRACT**(21) Appl. No.: **13/527,867**(22) Filed: **Jun. 20, 2012****Related U.S. Application Data**(60) Provisional application No. 61/499,121, filed on Jun.
20, 2011.

Systems and methods for maintaining data security using client roles, API keys, and certificate-based mutual authentication are presented. A method of protecting sensitive data includes both client authorization techniques and the mutual exchange and verification of certificates between client and server. In one embodiment, access by a client to a server is further limited by temporal constraints, volume constraints, and an end user identity filter.



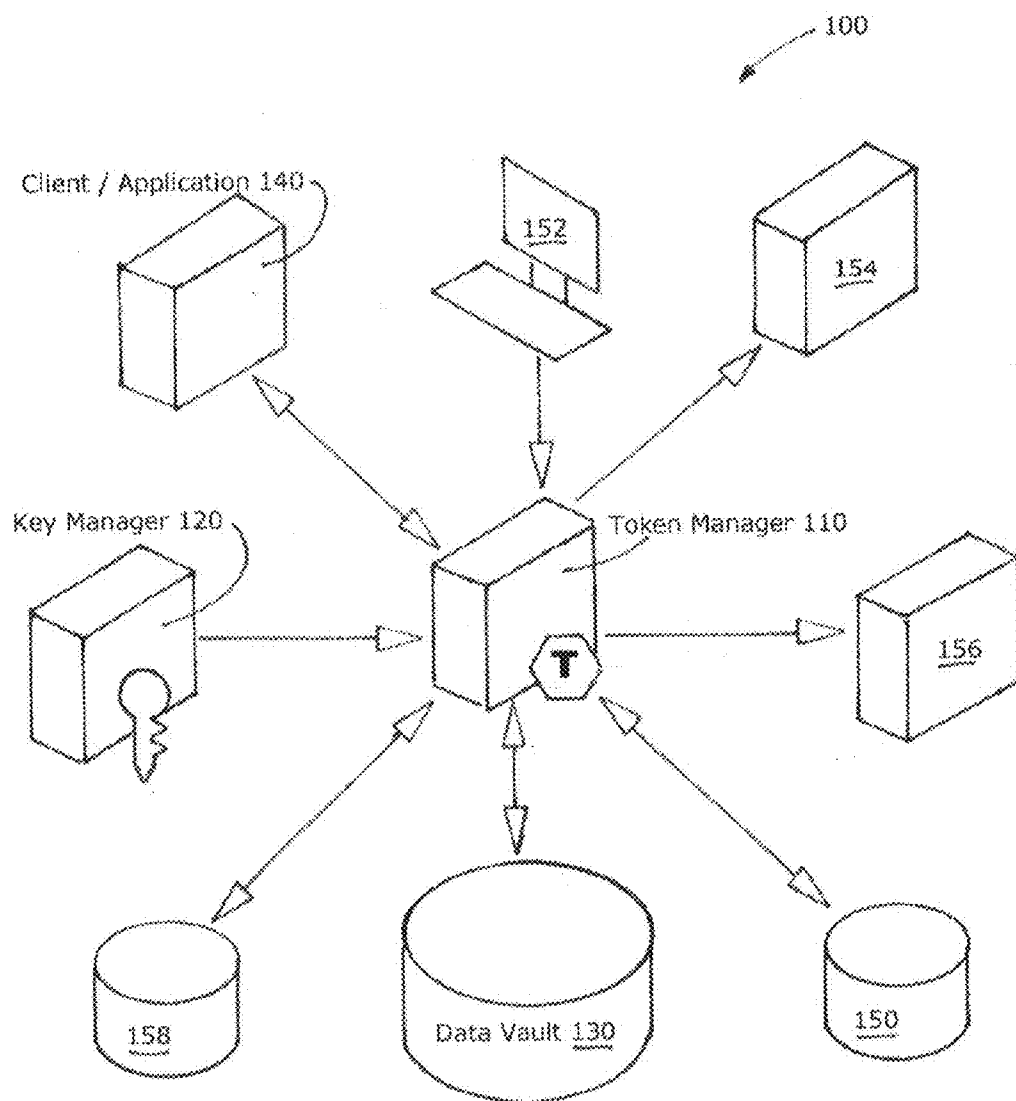


Fig. 1

Sensitive Data 10 →	375257	122500	3125
Token 200a →	375257	X4gmbAdL4Qz	3125
	Head	Body	Tail
	202a	204a	206a

Fig. 2A

Sensitive Data 10 →	375257	122500	3125
Token 200b →	375257	433902	3125
	Head	Body	Tail
	202b	204b	206b

Fig. 2B

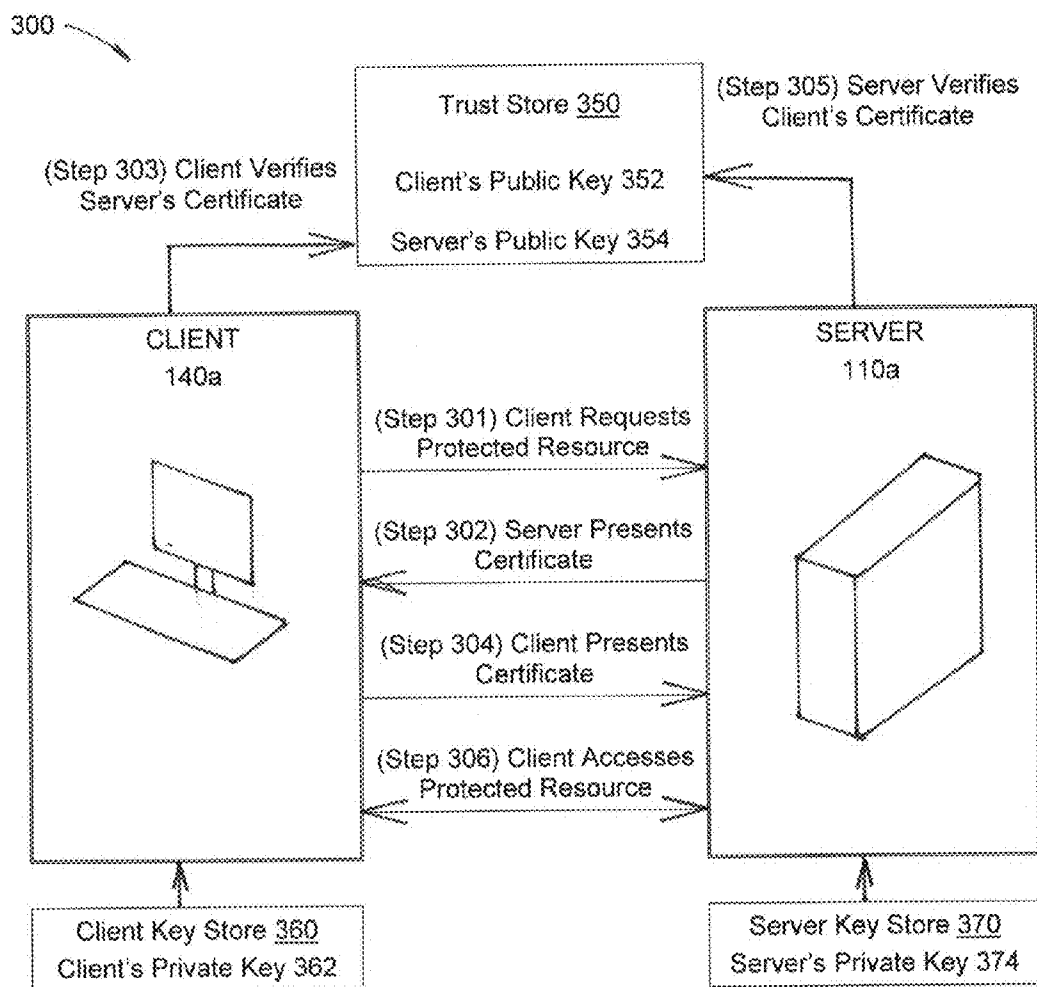


Fig. 3

CERTIFICATE-BASED MUTUAL AUTHENTICATION FOR DATA SECURITY

CROSS-REFERENCE TO RELATED APPLICATIONS

[0001] This application claims the benefit of and priority to U.S. Provisional Application No. 61/499,121, entitled "Token Manager for Data Protection," filed Jun. 20, 2011, which is herein incorporated by reference in its entirety.

TECHNICAL FIELD

[0002] The following disclosure relates generally to data security applications and, more particularly, to systems and methods for maintaining data security using certificate-based mutual authentication.

BACKGROUND

[0003] The proliferation of business-critical and sensitive electronic data creates a data security challenge, especially when sensitive data is collected in geographically distant locations and stored in multiple applications and databases for later processing.

[0004] Data encryption uses an encryption key to encrypt the sensitive data. The resulting encrypted data, sometimes called cipher text, can be stored in a database. The encrypted data is generally larger than the original value, requiring more space. Storing the encryption key in the same place exposes the encrypted data to easy decryption if the database is compromised.

[0005] Another layer of security is sometimes provided in the form of a token that represents or acts as a pointer to the encrypted data. Most existing token-based solutions require a centralized implementation with a single data store in order to minimize the risk of token collision, and to ensure a one-to-one relationship between a token and the sensitive data it represents.

[0006] Protecting sensitive data from unauthorized disclosure is a significant challenge, especially in computing environments with a large number and variety of client applications, devices, and end users. Increasingly complex systems require continuing advancement in the area of client access authorization and control.

SUMMARY

[0007] A method of protecting sensitive data from unauthorized access, according to various embodiments, includes the computer-implemented steps of: (1) establishing a plurality of client roles for authorizing access to a set of API functions; (2) establishing a plurality of API keys for authorizing access to a subset of the set of API functions; (3) establishing one or more constraints for limiting access to the subset of the set of API functions, the constraints comprising one or more temporal constraints, one or more volume constraints, and an end user identity filter; (4) assigning a first client role and a first API key to a first client; (5) receiving from the first client, at a first server, a request for access to a protected API function; (6) requiring the mutual exchange and verification of certificates between the first client and the first server; and (7) allowing the request if both the first client role and the first API key authorize access to the protected API function, while also limiting the access by imposing one or more of the one or more constraints.

[0008] In another aspect of the method, the one or more temporal constraints is a time-related limit selected from the group consisting of UTC clock time, client's local time of day, server's local time of day, client's local day of the week, and server's local day of the week. The one or more volume constraints comprises a limit on the quantity of requests by the first client within a predetermined time window. The end user identity filter comprises a rule permitting the display of masked values only unless an identified end user making the request matches a name on an approved user list.

BRIEF DESCRIPTION OF THE DRAWING

[0009] Having thus described various embodiments in general terms, reference will now be made to the accompanying drawings, which are not necessarily drawn to scale, and wherein:

[0010] FIG. 1 is an exemplary system architecture diagram, according to particular embodiments.

[0011] FIG. 2A is an illustration of sensitive data and a corresponding token, according to particular embodiments.

[0012] FIG. 2B is an illustration of sensitive data and a corresponding token, according to particular embodiments.

[0013] FIG. 3 is a diagram of an exemplary authentication system, according to particular embodiments.

DETAILED DESCRIPTION

[0014] The present systems and apparatuses and methods are understood more readily by reference to the following detailed description, examples, drawing, and claims, and their previous and following descriptions. However, before the present devices, systems, and/or methods are disclosed and described, it is to be understood that this invention is not limited to the specific devices, systems, and/or methods disclosed unless otherwise specified, as such can, of course, vary. It is also to be understood that the terminology used herein is for the purpose of describing particular aspects only and is not intended to be limiting.

[0015] The following description is provided as an enabling teaching in its best, currently known embodiment. To this end, those skilled in the relevant art will recognize and appreciate that many changes can be made to the various aspects described herein, while still obtaining the beneficial results of the technology disclosed. It will also be apparent that some of the desired benefits can be obtained by selecting some of the features while not utilizing others. Accordingly, those with ordinary skill in the art will recognize that many modifications and adaptations are possible, and may even be desirable in certain circumstances, and are a part of the invention described. Thus, the following description is provided as illustrative of the principles of the invention and not in limitation thereof.

[0016] As used throughout, the singular forms "a," "an" and "the" include plural referents unless the context clearly dictates otherwise. Thus, for example, reference to "a" component can include two or more such components unless the context indicates otherwise.

[0017] Ranges can be expressed herein as from "about" one particular value, and/or to "about" another particular value. When such a range is expressed, another aspect includes from the one particular value and/or to the other particular value. Similarly, when values are expressed as approximations, by use of the antecedent "about," it will be understood that the particular value forms another aspect. It will be further under-

stood that the endpoints of each of the ranges are significant both in relation to the other endpoint, and independently of the other endpoint.

[0018] As used herein, the terms “optional” or “optionally” mean that the subsequently described event or circumstance may or may not occur, and that the description includes instances where said event or circumstance occurs and instances where it does not.

Exemplary Tokenization System

[0019] Aspects of this disclosure relate to systems and methods for protecting and using sensitive data such as credit card numbers in compliance with regulations and best practices. Although the systems and methods are described herein primarily within the context of credit card numbers, the technology described herein is useful and applicable for protecting any type of sensitive data, such as social security numbers, passport numbers, license numbers, account numbers, payroll data, national health insurance numbers, personally-identifiable information (PII) such as name and date of birth, and the like.

[0020] FIG. 1 illustrates the architecture for an exemplary system 100, according to particular embodiments. The system 100 as shown includes four distinct modules: a token manager 110, a key manager 120, a data vault 130, and a client/application 140.

[0021] The key manager 120 manages encryption keys that are used to encrypt sensitive data and permit only authorized users to reveal or otherwise access the sensitive data. The encryption keys may be distributed to the token manager 110 for use in encryption and decryption functions.

[0022] The token manager 110 is a central part of the system 100, providing tokenization, encryption, client management, event logging, and administrative functions. Tokenization describes the process of receiving sensitive data and generating a token to be used in its place. The token manager 110 generates the token, encrypts the original sensitive data, and stores the encrypted data (cipher text) in the data vault 130. The encrypted data is stored only in the data vault 130. The token is a reference to the encrypted data; there is no mathematical relationship between a token and the encrypted data. Therefore, the token may be safely used throughout the system 100, while the encrypted data it represents remains stored in the data vault 130. The token manager 110 ensures that there is a one-to-one relationship between the sensitive data and the generated token, so that referential integrity is maintained throughout the system 100.

[0023] The data vault 130 is a depository such as a database for storing the tokens and the encrypted data. The data vault does not store the encryption key, which is stored and controlled using the key manager 120. In particular embodiments, the data vault 130 may store a key profile number or other pointer that indicates which key was used to encrypt the data. The token manager 110 may use a data access technology such as JDBC (Java Database Connectivity) to communicate with the data vault 130.

[0024] The client or application 140 may be any of a variety of applications or platforms involved in the collection, handling, or processing of sensitive data. For example, the client/application 140 may be a financial application for processing or analyzing payments received by a business enterprise. Another client/application 140 may be a point-of-sale device such as a cash register or payment card reader. In particular embodiments, integration of client/applications 140 may be

accomplished through SOAP/web services. In this aspect, any application 140 that supports web services can be integrated with the token manager 110 and may be configured to make calls to tokenize/encrypt sensitive data or, if authorized, to decrypt/access the sensitive data.

[0025] As illustrated in FIG. 1, the system 100 may include other modules depending on the implementation. For example, the system 100 may include a directory 150 includes a database for storing any type of data useful in the system 100. For example, the directory 150 may include client IP addresses, hostnames, user identities, client role definitions, client permissions and data access policies, and the like. The token manager 110 may use LDAP or another protocol for accessing and maintaining the directory 150.

[0026] The system 100 may also include an administrator 152 with access to the token manager 110. The administrator 152 may use HTTP/S or another secure protocol for communicating with the token manager 110.

[0027] The token manager 110 and the key manager 120 may be configured to generate security event messages via Syslog. These logs can be directed to an event log 154 which may include an event management application (SIEM) for logging, collecting, storing, analyzing, and/or reporting events.

[0028] The token manager 110 may also be configured to send e-mail alerts using an e-mail server 156 via SMTP or similar protocol. The system 100 may also include a metadata store 158.

[0029] In use, the token manager 110, according to particular embodiments, receives sensitive data from an application 140 along with a request to protect it, generates a token, and inserts the token in place of the sensitive data. The key manager 120 supplies an encryption key to the token manager 110, which then encrypts the sensitive data and stores the encrypted data (cipher text) in the data vault 130. Tokens can be used safely in any application or database without exposing the sensitive data.

[0030] When an application 140 or database requires the original sensitive data, the application 140 transmits a request (by web services call, for example) to the token manager 110 and presents the token. The token manager 110 validates the credentials of the requesting application and, if authorized, looks-up the token in the data vault 130, identifies the matching cipher text, decrypts the cipher text, and returns the original sensitive data back to the application 140.

[0031] According to particular embodiments, the system 100 ensures a one-to-one relationship between a token and the sensitive data it represents. The data vault 130 contains a single encrypted version of each original sensitive data. Even when encryption keys change over time, there is only one instance of the encrypted value stored in the data vault 130. In use, this means that the returned token will consistently represent the same original data throughout the system 100, in different applications and across multiple data sets.

Token Characteristics

[0032] The token manager 110 in particular embodiments may be configured to generate a token that is structurally similar in length and format to that of the original sensitive data. For example, as shown in FIG. 2A, a token 200a can be formatted to preserve any number of leading and trailing characters found in the original sensitive data 10. In the example shown, the head 202a includes the leading six char-

acters, the tail **206a** includes the trailing four, and the body **204a** includes tokenized characters.

[0033] As shown in FIG. 2B, a token **200b** can be formatted to preserve both the length and the data type (alpha or numeric, and the like) of the original sensitive data **10**. In the example shown, the head **202b** includes the leading six characters, the body **204b** includes six numeric characters, and the tail **206b** includes the trailing four characters. Any number (including zero) of the leading and/or trailing characters from the original sensitive data **10** may be preserved. The format-preserving tokenization process is also described in the commonly owned and co-pending U.S. patent application Ser. No. 13/040,133, entitled "System and Methods for Format Preserving Tokenization of Sensitive Information," which is herein incorporated by reference in its entirety.

[0034] The task of generating a token may be accomplished in one or more steps performed by a token generation algorithm. As described above, the token generation algorithm may be configured to generate a token that is independent of at least a portion of the data in a sensitive data string.

Client Access Control and Authorization

[0035] The token manager **110** in particular embodiments requires authentication at multiple levels in order to provide secure access control. Referring again to FIG. 1, the administrator **152** in particular embodiments manages client authentication and authorization through the creation, control, and maintenance of Clients, Client Roles, API Keys, and Mutual Authentication.

[0036] Clients:

[0037] A Client accesses the services provided by the token manager **110** or other server through SOAP/web services API calls. Through a user interface called a management console, the administrator **152** may create client identities, including the client's IP address/hostname and other identifying characteristics. The administrator **152** may enable or disable a Client at any time via the management console for any reason; for example, to protect sensitive data. Referring briefly to FIG. 3, a Client **140a** requests access to the services provided by the token manager or other server **110a**.

[0038] Client Roles:

[0039] All Clients are assigned to a Client Role. A Client can only belong to a single Client Role; however, a Client Role may include multiple Clients. Client Roles are used to define the rights and restrictions of all the member Clients. Because the creation of authorization policies is done at the Client Role level, rather than at the Client level, administration of those policies is simplified (even for a large number of Clients and Client Roles). Every member Client is bound by the rules of the Client Role. For example, a Client Role may be established for the Finance Department of a business enterprise, with permissions granted to access a particular set of API functions. All Clients associated with any applications used by the Finance Department would be assigned to this Client Role.

[0040] API Keys:

[0041] An API (Application Programming Interface) is an interface that facilitates communication between various software components. An API may include routines, data structures, object classes, variables, and the like. In particular embodiments, for example, the token manager described herein may include API functions such as protect (i.e., protect sensitive data using encryption and/or tokenization), protect bulk (i.e., multiple items of sensitive data in a set), reveal (i.e.,

decrypt and expose the sensitive data), reveal bulk, send to wastebasket, send to wastebasket bulk, retrieve from wastebasket, retrieve from wastebasket bulk, empty wastebasket, purge tokens, and lookup.

[0042] An API key is a random string that is known to both the Client and the token manager or other server providing resources. When calling an API function, the Client must provide the correct API Key. Authorization based on API Keys can help differentiate permissions between multiple Clients coming from the same host and/or multiple Clients being passed through a proxy service. Continuing the example above, a number of API Keys must be established for the Finance Department. For example, an API Key for a finance application should allow access to only those API functions that are called by that finance application. A separate API Key for an auditing application should allow access to only those API functions that are called by the auditing application. In this way, the API Keys provide another level of granularity to the client authorization process.

[0043] API Keys are used in tandem with Client Roles; in other words, the Client must have permission from both its Client Role and its API Key in order to access a given API function on a token manager or other server. If the Client does not present both a Client Role permission and a valid API Key, the Client's request will be rejected.

[0044] For example, a Client Role may be established for point-of-sale devices. Client processes and applications assigned to this Client Role may be given authorization to "protect" and to "reveal" sensitive data (such as a credit card number). Certain types of users of the point-of-sale devices may be further controlled by using API Keys. A teller, for example, may be assigned an API Key that authorizes access to only the "protect" API function. A store manager may be assigned a different API Key, for example, that authorizes access to both the "protect" and "reveal" API functions. In use, a teller using a point-of-sale device who attempts to access the "reveal" API (and thereby obtain sensitive data, such as a credit card number) will be rejected. A store manager, however, can use the same point-of-sale device and will be allowed to access the "reveal" API function because her API Key allows it (to process a refund, for example).

[0045] Client Role Authorization Policies:

[0046] Authorization policies are customizable for each Client Role by the administrator **152**. The administrator **152** can establish, revise, and update a wide variety of permissions and constraints. For example, the administrator **152** can assign a Client to a particular Client Role. The administrator **152** can vary the API functions to which a Client Role has access. The administrator **152** can establish the types of data allowed for access or presentation to certain Clients. Client role authorization can also be used to create permissions based on IP address, a range of IP addresses, hostname, and the like.

[0047] Authorizations may be based on temporal constraints, imposed alone or in combination with others. Temporal constraints, for example, may include the UTC clock time, the local time of day (at the client, server, or another location), the day of the week, the day of the month, week of the year, quarter of the year, or any other time- or date-based limitation. The local time of day and/or day of the week can be particularly useful in retail and other settings where the hours of operation (for retail and other processing operations) are

generally known. A teller may be authorized only during retail hours, whereas a payment processor may be authorized during overnight hours.

[0048] Authorizations may also include volume constraints, such as a limit on the number of API calls made from a particular device or during a certain time window.

[0049] End User Identity:

[0050] A request to access and use the “reveal” API function (which exposes sensitive data, such as a credit card number) in particular embodiments requires confirmation of the end user’s personal identity. User-level authorizations can determine if sensitive data should be returned as clear text or as partially masked text (masking digits with asterisks, for example).

[0051] Certificate-Based Mutual Authentication,

[0052] as described more fully below, involves a mutual exchange of certificates between Client 140a and Server 110a, in particular embodiments, before the Server will grant access to the API functions or other protected resources provided by the Server.

[0053] The tools described herein for client access and authentication may be supported by information in the directory 150 (shown in FIG. 1) which, in particular embodiments, may be accessed using LDAP or another suitable protocol. The management console used by the administrator 152 may include access to the directory 150 in order to create, read, update, or delete information about Clients such as IP addresses and hostnames, Client Role definitions and authorization policies, API Keys, end user identities, certificates and keys, and the like.

Mutual Authentication

[0054] FIG. 3 is a diagram illustrating an exemplary mutual authentication system 300, according to particular embodiments. The Server 110a may be a token manager or other server configured to provide resources to one or more Clients 140a. A trust store 350 is a repository for storing the public key portion of a cryptographic system that requires two separate keys; a public key and a private key. RSA is an algorithm often used in two-key cryptography. The keys correspond to trusted entities.

[0055] The client key store 360 is a repository for storing the client’s private key 362. The server key store 370 is a repository for storing the server’s private key 374.

[0056] FIG. 3 illustrates the steps involved in a mutual authentication, according to particular embodiments. First, in Step 301, the Client 140a sends a request to a Server 110a for a protected resource. In response the Server presents its certificate (Step 302). In Step 303, the Client accesses the trust store 350 to retrieve the server’s public key 354 and thereby verify the server’s certificate. Next, the Client 140a presents its own certificate to the server (Step 304). In Step 305, the Server accesses the trust store 350 to retrieve the client’s public key 352 and thereby verify the client’s certificate. These verification steps accomplished by both Client and Server make this a mutual authentication system. Once the mutual authentication is completed, the Client may access the protected resources offered by the Server (Step 306).

Client Authorization and Mutual Authentication

[0057] In particular embodiments, the token manager or other Server 110a requires both client authorization and mutual authentication before permitting client access. In one

embodiment, at the server level, a method of protecting sensitive data includes the step of establishing a plurality of Client Roles for authorizing access to a set of API functions, establishing a plurality of API Keys for authorizing access to a subset of the set of API functions, establishing one or more constraints for limiting access to the subset, and requiring the mutual exchange and verification of certificates between the client and server before a request for access is allowed.

[0058] The Client Roles, as described herein, include a number of parameters defining the scope and authority of member Clients. At a more granular level, the API Keys include a number of parameters further defining the scope and authority of Clients and applications that request access to certain API functions; typically, a subset of the set of API functions authorized at the Client Role level.

[0059] The additional constraints may include one or more temporal constraints, imposed either alone or in combination. A volume constraint may include a limit on the number of requests made from a particular device, or during a predetermined time window, such as a range of hours, a entire day, or a particular day of the week. An end user constraint may include an end user identity filter. If an identified end user’s name is found on an approved user list, then access to the requested API function may be unlimited; otherwise, access is limited according to the user’s identity. For example, for an approved end user, an API function may return sensitive data in the form of clear text, whereas an unapproved user may only be shown partially masked text.

[0060] Any of the client authorization techniques, together with the mutual authentication regimen, may be combined in a method of protecting sensitive data.

CONCLUSION

[0061] Although the systems and methods are described herein primarily within the context of numerical data such as credit card numbers, the technology described herein is useful and applicable for protecting any type of sensitive data, such as social security numbers, passport numbers, license numbers, account numbers, payroll data, national health insurance numbers, personally-identifiable information (PII) such as name and date of birth, and the like. Moreover, although several embodiments have been described herein, those of ordinary skill in art, with the benefit of the teachings of this disclosure, will understand and comprehend many other embodiments and modifications for this technology. The invention therefore is not limited to the specific embodiments disclosed or discussed herein, and that may other embodiments and modifications are intended to be included within the scope of the appended claims. Moreover, although specific terms are occasionally used herein, as well as in the claims or concepts that follow, such terms are used in a generic and descriptive sense only, and should not be construed as limiting the described invention or the claims that follow.

1. A method of protecting sensitive data from unauthorized access, said method comprising the computer-implemented steps of:

- establishing a plurality of client roles for authorizing access to a set of API functions;
- establishing a plurality of API keys for authorizing access to a subset of said set of API functions;
- providing a first client role and a first API key to a first client;

receiving from said first client, at a first server, a request for access to a protected API function;

requiring the mutual exchange and verification of certificates between said first client and said first server; and allowing said request if both said first client role and said first API key authorize access to said protected API function.

2. The method of claim 1, further comprising the step of limiting access to said set of API functions by imposing one or more temporal constraints, one or more volume constraints, and/or an end user identity filter.

3. The method of claim 2, wherein said one or more temporal constraints is a time-related limit selected from the group consisting of UTC clock time, client's local time of day, server's local time of day, client's local day of the week, and server's local day of the week.

4. The method of claim 2, wherein said one or more volume constraints comprises a limit on the quantity of requests by said first client within a predetermined time window.

5. The method of claim 2, wherein said end user identity filter comprises a rule permitting the display of masked values only unless an identified end user making said request matches a name on an approved user list.

6. A method of protecting sensitive data from unauthorized access, said method comprising the computer-implemented steps of:

establishing a plurality of client roles for authorizing access to a set of API functions;

establishing a plurality of API keys for authorizing access to a subset of said set of API functions;

establishing one or more constraints for limiting access to said subset of said set of API functions, said constraints comprising one or more temporal constraints, one or more volume constraints, and an end user identity filter; assigning a first client role and a first API key to a first client;

receiving from said first client, at a first server, a request for access to a protected API function;

requiring the mutual exchange and verification of certificates between said first client and said first server; and allowing said request if both said first client role and said first API key authorize access to said protected API function, while also limiting said access by imposing one or more of said one or more constraints.

7. The method of claim 6, wherein said one or more temporal constraints is a time-related limit selected from the group consisting of UTC clock time, client's local time of day, server's local time of day, client's local day of the week, and server's local day of the week.

8. The method of claim 6, wherein said one or more volume constraints comprises a limit on the quantity of requests by said first client within a predetermined time window.

9. The method of claim 6, wherein said end user identity filter comprises a rule permitting the display of masked values only unless an identified end user making said request matches a name on an approved user list.

* * * * *