DATA SECURITY SERVICE

CROSS-REFERENCE TO RELATED APPLICATIONS

[0001] This application claims priority to U.S. Patent Application No. 13/764,963, filed February 12, 2013, the contents of which are incorporated herein by reference in its entirety. This application incorporates by reference for all purposes the full disclosure of co-pending U.S. Patent Application No. 13/764,944, filed concurrently herewith, entitled "AUTOMATIC KEY ROTATION", co-pending U.S. Patent Application No. 13/764,995, filed concurrently herewith, entitled "POLICY ENFORCEMENT WITH ASSOCIATED DATA", co-pending U.S. Patent Application No. 13/765,020, filed concurrently herewith, entitled "DATA SECURITY WITH A SECURITY MODULE", co-pending U.S. Patent Application No. 13/765,209, filed concurrently herewith, entitled "FEDERATED KEY MANAGEMENT", co-pending U.S. Patent Application No. 13/765,239, filed concurrently herewith, entitled "DELAYED DATA ACCESS", co-pending U.S. Patent Application No. 13/765,265, filed concurrently herewith, entitled "DATA SECURITY SERVICE", and co-pending U.S. Patent Application No. 13/765,283, filed concurrently herewith, entitled "SECURE MANAGEMENT OF INFORMATION USING A SECURITY MODULE".

BACKGROUND

[0002] The security of computing resources and associated data is of high importance in many contexts. As an example, organizations often utilize networks of computing devices to provide a robust set of services to their users. Networks often span multiple geographic boundaries and often connect with other networks. An organization, for example, may support its operations using both internal networks of computing resources and computing resources managed by others. Computers of the organization, for instance, may communicate with computers of other organizations to access and/or provide data while using services of another organization. In many instances, organizations configure and operate remote networks using hardware managed by other organizations, thereby reducing infrastructure costs and achieving other advantages. With such configurations of computing resources, ensuring that access to the resources and the data they hold is secure can be challenging, especially as the size and complexity of such configurations grow.

BRIEF DESCRIPTION OF THE DRAWINGS

[0003] Various embodiments in accordance with the present disclosure will be described with reference to the drawings, in which:

[0004] FIG. 1 shows an illustrative diagram representing various aspects of the present disclosure in accordance with various embodiments;

[0005] FIG. 2 shows an illustrative example of an environment in which various aspects of the present disclosure may be implemented;

[0006] FIG. 3 shows an illustrative example of an environment in which various aspects of the present disclosure may be implemented and an example flow of information among the various components of the environment in accordance with at least one embodiment;

[0007] FIG. 4 shows example steps of an illustrative process for storing ciphertext, in accordance with at least one embodiment;

[0008] FIG. 5 shows an illustrative example of an environment in which various aspects of the present disclosure may be implemented and an example flow of information among the various components of the environment in accordance with at least one embodiment;

[0009] FIG. 6 shows example steps of an illustrative process for responding to a request to retrieve data, in accordance with at least one embodiment;

[0010] FIG. 7 shows an illustrative example of an environment in which various aspects of the present disclosure may be implemented and an example flow of information among the various components of the environment in accordance with at least one embodiment;

[0011] FIG. 8 shows example steps of an illustrative process for responding to a request to store data, in accordance with at least one embodiment;

[0012] FIG. 9 shows an illustrative example of an environment in which various aspects of the present disclosure may be implemented and an example flow of information among the various components of the environment in accordance with at least one embodiment;

[0013] FIG. 10 shows example steps of an illustrative process for responding to a request to retrieve data, in accordance with at least one embodiment;

[0014] FIG. 11 shows an illustrative example of an environment in which various aspects of the present disclosure may be implemented;

[0015] FIG. 12 shows an illustrative example of an environment in which various aspects of the present disclosure may be implemented and an example flow of information among the various components of the environment in accordance with at least one embodiment;

[0016] FIG. 13 shows example steps of an illustrative process for responding to a request to retrieve data, in accordance with at least one embodiment;

[0017] FIG. 14 shows example steps of an illustrative process for responding to a request to decrypt data, in accordance with at least one embodiment;

[0018] FIG. 15 shows example steps of an illustrative process for obtaining decrypted data, in accordance with at least one embodiment;

[0019] FIG. 16 shows a diagrammatic representation of an example cryptography service, in accordance with at least one embodiment;

[0020] FIG. 17 shows example steps of an illustrative process for configuring policy, in accordance with at least one embodiment;

[0021] FIG. 18 shows example steps of an illustrative process for performing cryptographic operations while enforcing policy, in accordance with at least one embodiment; and

[0022] FIG. 19 shows an illustrative example of an environment in which various embodiments may be implemented.

DETAILED DESCRIPTION

[0023] In the following description, various embodiments will be described. For purposes of explanation, specific configurations and details are set forth in order to provide a thorough understanding of the embodiments. However, it will also be apparent to one skilled in the art that the embodiments may be practiced without the specific details. Furthermore, well-known features may be omitted or simplified in order not to obscure the embodiment being described.

[0024] Techniques described and suggested herein allow for enhanced data security in environments involving distributed computing resources. In one example, a distributed computing environment includes one or more data services which may be implemented by appropriate computing resources. The data services may allow various operations to be performed in connection with data. As one illustrative example, the distributed computing environment includes one or more data storage services. Electronic requests may be

transmitted to the data storage service to perform data storage operations. Example operations are operations to store data using the data storage service and using the data storage service to retrieve data stored by the data storage service. Data services, including data storage services, may also perform operations that manipulate data. For example, in some embodiments, a data storage service is able to encrypt data.

[0025] Various embodiments of the present disclosure include distributed computing environments that include cryptography services that are implemented using appropriate computing resources. A cryptography service may be implemented by a distributed system that receives and responds to electronic requests to perform cryptographic operations, such as encryption of plaintext and decryption of ciphertext. In some embodiments, a cryptography service manages keys. In response to a request to perform a cryptographic operation, the cryptography service may execute cryptographic operations that use the managed keys. For example, the cryptography service can select an appropriate key to perform the cryptographic operation, perform the cryptographic operation, and provide one or more results of the cryptographic operation in response to the received request. In an alternative configuration, the cryptography service can produce an envelope key (e.g., a session key that is used to encrypt specific data items) and return the envelope key to the system invoking the cryptographic operations of the service. The system can then use the envelope key to perform the cryptographic operation.

[0026] In some embodiments, the cryptography service manages keys for multiple tenants of a computing resource service provider. A tenant of the computing resource may be an entity (e.g. organization or individual) operating as a customer of the computing resource provider. The customer may remotely and programmatically configure and operate resources that are physically hosted by the computing resource provider. When a customer submits a request to the cryptography service to perform a cryptographic operation (or when an entity submits a request to the cryptography service), the cryptography service may select a key managed by the cryptography service for the customer to perform the cryptographic operation. The keys managed by the cryptography service may be securely managed so that other users and/or data services do not have access to the keys of others. A lack of access by an entity (e.g., user, customer, service) to the key of another entity may mean that the entity does not have an authorized way of obtaining the key of the other and/or that the entity does not have an authorized way of causing a system that manages the key of the other of using the key at the entity's direction. For example, the cryptography service may manage keys so that, for a customer, other customers both do not have access to the customer's key(s) and are

unable to cause the cryptography service to use the customer's key(s) to perform cryptographic operations. As another example, the cryptography service may manage keys so that other services, such as a data storage service, are unable to cause the cryptography service to use some or all keys to perform cryptographic operations. Unauthorized access to a key may be prevented by appropriate security measures so that, for example, unauthorized access is difficult or impossible. Difficulty may be due to computational impracticality and/or due to a need for an unauthorized (e.g., illegal, tortious and/or otherwise disallowed such as a. compromise of authorization credentials) to occur for access to be gained. Systems in accordance with the various embodiments may be configured to ensure an objective measure of computational impracticality to gain access to a key. Such measures may be, for example, measured in terms of an amount of time, on average, it would take a computer having a defined unit of computational ability (e.g. certain operations per unit of time) to crack encrypted information needed for authorized access to the key.

[0027] As noted, a cryptography service may receive requests from various entities, such as customers of a computing resource provider. A cryptography service may also receive requests from entities internal to the computing resource provider. For example, in some embodiments, data services implemented by the computing resource provider may transmit requests to a cryptography service to cause the cryptography service to perform cryptographic operations. As one example, a customer may transmit a request to a data storage service to store a data object. The request may indicate that the data object should be encrypted when stored. The data storage service may communicate a request to a cryptography service to perform a cryptographic operation. The cryptographic operation may be, for example, encryption of a key used by the data storage service to encrypt the data object. The cryptographic operation may be encryption of the data object itself. The cryptographic operation may be to generate an envelope key that the data storage service can use to encrypt the data object.

[0028] Systems in accordance with the various embodiments implement various security measures to provide enhanced data security. For example, in various embodiments, the manner in which a cryptography service can utilize keys it manages is limited. For example, in some embodiments, a cryptography service is configured to only use a key corresponding to a customer upon appropriate authorization. If a request to use a customer's key purportedly originates from the customer (i.e., from a computing device operating on behalf of the customer), the cryptography service may be configured to require that the request be electronically (digitally) signed using appropriate credentials owned by the customer. If the

request to use the customer's key originated from another data service, the cryptography service may be configured to require that the data service provide proof that a signed request to the data service has been made by the customer. In some embodiments, for example, the data service is configured to obtain and provide a token that serves as proof of an authenticated customer request. Other security measures may also be built into a configuration of an electronic environment that includes a cryptography service. For example, in some embodiments, a cryptography service is configured to limit key use according to context. As one illustrative example, the cryptography service may be configured to use a key for encryption for requests from a customer or from a data service acting on behalf of the customer. However, the cryptography service may be configured to only use a key for decryption for requests from the customer (but not from another data service). In this manner, if the data service is compromised, the data service would not be able to cause the cryptography service to decrypt data.

[0029] Various security measures may be built into a cryptography service and/or its electronic environment. Some security measures may be managed according to policies which, in some embodiments, are configurable. As one example, a cryptography service may utilize an application programming interface (API) that enables users to configure policies on keys. A policy on a key may be information that, when processed by a cryptography service, is determinative of whether the key may be used in certain circumstances. A policy may, for instance, limit identities of users and/or systems able to direct use of the key, limit times when the key can be used, limit data on which the key can be used to perform cryptographic operations on, and provide other limitations. The policies may provide explicit limitations (e.g., who cannot use the keys) and/or may provide explicit authorizations (e.g., who can use the keys). Further, policies may be complexly structured to generally provide conditions for when keys can and cannot be used. When a request to perform a cryptographic operation using a key is received, any policies on the key may be accessed and processed to determine if the request can, according to policy, be fulfilled.

[0030] FIG. 1 is an illustrative diagram 100 demonstrating various embodiments of the present disclosure. In an embodiment, a cryptography service performs cryptographic operations which may include application of one or more calculations in accordance with one or more cryptographic algorithms. As illustrated in FIG. 1, the cryptography service enables a user or a service to generate plaintext from ciphertext. In an example configuration the cryptographic service can be used to encrypt/decrypt keys and these keys can be used to encrypt/decrypt data, such as data stored in a data storage service. For example, the

cryptography service receives a request to generate plaintext from ciphertext encrypted under a key. The cryptography service determines that the requestor is an authorized entity; decrypts the key using a master key and returns the now decrypted key to the service, which can generate plaintext from the ciphertext using the decrypted key. In another configuration, the cryptography service receives ciphertext and processes the received ciphertext into plaintext which is provided as a service by the cryptography service. In this example, the ciphertext may be provided to the cryptography service as part of an electronic request to the cryptography service from an authorized entity, which may be a customer of a computing resource provider that operates the cryptography service and/or which may be another service of the computing resource provider. The cryptography service illustrated in FIG. 1 may utilize one or more cryptographically strong algorithms to encrypt data. Such cryptographically strong algorithms may include, for example, Advanced Encryption Standard (AES), Blowfish, Data Encryption Standard (DES), Triple DES, Serpent or Twofish, and depending on the specific implementation selected, may be either asymmetric or symmetric key systems. Generally, the cryptography service may utilize any encryption and/or decryption algorithm (cipher) or combination of algorithms that utilizes data managed by the cryptography service.

[0031] As will be discussed below in more detail, the cryptography service can be implemented in a variety of ways. In an embodiment, the cryptography service is implemented by a computer system configured in accordance with the description below. The computer system may itself comprise one or more computer systems. For example, a cryptography service may be implemented as a network of computer systems collectively configured to perform cryptographic operations in accordance with the various embodiments. Or put another way, the computer system could be a distributed system. In an embodiment, ciphertext is information that has been encrypted using a cryptographic algorithm. In the example of FIG. 1, the ciphertext is the plaintext in an encrypted form. The plaintext may be any information and while the name includes no word text, plaintext and ciphertext may be information encoded in any suitable form and does not necessarily include textual information but it may include textual information. For example, as illustrated in FIG. 1, plan text and ciphertext comprise sequences of bits. Plaintext and ciphertext may also be represented in other ways and generally in any manner by which encryption and decryption can be performed by a computer system.

[0032] FIG. 2 shows an illustrative example of an environment 200 in which a cryptography service, such as illustrated in FIG. 1, may be implemented. In the environment

of 200, various components operate together in order to provide secure data related services. In this particular example, the environment 200 includes a cryptography service, an authentication service, a data service frontend and a data service backend storage system. In an embodiment, the cryptography service is configured in the environment 200 to perform cryptographic operations, such as by receiving plaintext from a data service frontend and providing ciphertext in return or providing envelope keys to services, so that the services can use the envelope keys to perform encryption operations. The cryptography service may perform additional functions, such as described below, such as secure storage of keys for performance of cryptographic operations, such as converting plaintext into ciphertext and decrypting ciphertext into plaintext. The cryptography service may also perform operations involved in policy enforcement, such as by enforcing policy associated with keys stored therein. Example policies that may be enforced by a cryptography service are provided below. The data service frontend in an embodiment is a system configured to receive and respond to requests transmitted over a network from various users. The requests may be requests to perform operations in connection with data stored or to be stored in the data service backend storage system. In the environment 200, the authentication service, cryptography service, data service frontend and data service backend storage system may be systems of a computing resource provider that utilizes the systems to provide services to customers represented by the users illustrated in FIG. 2. The network illustrated in FIG. 2 may be any suitable network or combination of networks, including those discussed below.

[0033] The authentication service in an embodiment is a computer system configured to perform operations involved in authentication of the users. For instance, the data service frontend may provide information from the users to the authentication service to receive information in return that indicates whether or not the user requests are authentic. Determining whether user requests are authentic may be performed in any suitable manner and the manner in which authentication is performed may vary among the various embodiments. For example, in some embodiments, users electronically sign messages transmitted to the data service frontend. Electronic signatures may be generated using secret information (e.g., a private key of a key pair associated with a user) that is available to both an authenticating entity (e.g., user) and the authentication service. The request and signatures for the request may be provided to the authentication service which may, using the secret information, compute a reference signature for comparison with the received signature to determine whether the request is authentic. If the request is authentic, the authentication service may provide information that the data service frontend can use to prove to other services, such as the cryptography service, that the request is authentic, thereby enabling the

other services to operate accordingly. For example, the authentication service may provide a token that another service can analyze to verify the authenticity of the request. Electronic signatures and/or tokens may have validity that is limited in various ways. For example, electronic signatures and/or tokens may be valid for certain amounts of time. In one example, electronic signatures and/or tokens are generated based at least in part on a function (e.g., a Hash-based Message Authentication Code) that takes as input a timestamp, which is included with the electronic signatures and/or tokens for verification. An entity verifying a submitted electronic signature and/or token may check that a received timestamp is sufficiently current (e.g. within a predetermined amount of time from a current time) and generate a reference signature/token using for the received timestamp. If the timestamp used to generate the submitted electronic signature/token is not sufficiently current and/or the submitted signature/token and reference signature/token do not match, authentication may fail. In this manner, if an electronic signature is compromised, it would only be valid for a short amount of time, thereby limiting potential harm caused by the compromise. It should be noted that other ways of verifying authenticity are also considered as being within the scope of the present disclosure.

[0034] The data service backend storage system in an embodiment is a computer system that stores data in accordance with requests received through the data service frontend. As discussed in more detail below, the data service backend storage system may store data in encrypted form. Data in the data service backend storage system may also be stored in unencrypted form. In some embodiments, an API implemented by the data service frontend allows requests to specify whether data to be stored in the data service backend storage system should be encrypted. Data that is encrypted and stored in the data service backend storage system may be encrypted in various ways in accordance with the various embodiments. For example, in various embodiments, data is encrypted using keys accessible to the cryptography service, but inaccessible to some or all other systems of the environment 200. Data may be encoded by the cryptography service for storage in the data service backend storage system and/or, in some embodiments, data may be encrypted by another system, such as a user system or a system of the data service frontend, using a key that was decrypted by the cryptography service. Examples of various ways by which the environment 200 may operate to encrypt data are provided below.

[0035] Numerous variations of the environment 200 (and other environments described herein) are considered as being within the scope of the present disclosure. For example, the environment 200 may include additional services that may communicate with the

cryptography service and/or authentication service. For example, the environment 200 may include additional data storage services (which may each comprise a frontend system and a backend system) which may store data in different ways. For instance, one data storage service may provide active access to data where the data storage service performs data storage services in a synchronous manner (e.g., a request to retrieve data may receive a synchronous response with the retrieved data). Another data storage service may provide archival data storage services. Such an archival data storage service may utilize asynchronous request processing. For example, a request to retrieve data may not receive a synchronous response that includes the retrieved data. Rather, the archival data storage service may require a second request to be submitted to obtain the retrieved data once the archival data storage service is ready to provide the retrieved data. As another example, the environment 200 may include a metering service which receives information from the cryptography service (and/or other services) and uses that information to produce accounting records. The accounting records may be used to bill customers for usage of the cryptography service (and/or other services). Further, information from the cryptography service may provide an indication of how charges should be incurred. For instance, in some instances customers may be provided bills for use of the cryptography service. In other instances, charges for use of the cryptography service may be rolled into usage charges of other services, such as a data service that utilizes the cryptography service as part of its operations. Usage may be metered and billed in various ways, such as per operation, per period of time, and/or in other ways. Other data services may also be included in the environment 200 (or other environments described herein).

[0036] In addition, FIG. 2 depicts users interacting with the data service frontend. It should be understood that users may interact with the data service frontend through user devices (e.g. computers) which are not illustrated in the figure. Further, the users depicted in FIG. 2 (and elsewhere in the figures) may also represent non-human entities. For example, automated processes executing on computer systems may interact with the data service frontend as described herein. As one illustrative example, an entity represented by a user in FIG. 2 may be a server that, as part of its operations, uses the data service frontend to store and/or retrieve data to/from the data service backend storage system. As yet another example, an entity represented by a user in FIG. 2 may be an entity provided as a service of a computing resource provider that operates one or more of the services in FIG. 2. For instance, a user in FIG. 2 may represent a virtual or other computer system of a program execution service offered by the computing resource provider. Other variations, including variations of other

environments described below, are also considered as being within the scope of the present disclosure.

[0037] For example, FIG. 3 shows an illustrative example of an environment 300 in which various embodiments of the present disclosure may be implemented. As with FIG. 2, the environment in FIG. 3 includes an authentication service, a data service frontend system (data service front end), a cryptography service and a data service backend storage system. The authentication service, data service frontend, cryptography service and data service backend storage system may be configured such as described above in connection with FIG. 2. For example, users may access the data service frontend through a suitable communications network, although such network is not illustrated in the figure. In the example environment 300 illustrated in FIG. 3, arrows representing a flow of information are provided. In this example, a user transmits a PUT request to the data service frontend. The PUT request may be a request to store specified data in the data service backend storage system. In response to the PUT request, the data service frontend may determine whether the PUT request is authentic, that is if the user has submitted the request in the manner the requested operation can be performed in accordance with authentication policy implemented by the system.

[0038] In FIG. 3, an illustrative example of how such authentication decisions may be made is illustrated. In this particular example, the data service frontend submits an authentication request to the authentication service. The authentication service may use the authentication request to determine if the PUT request from the user is authentic. If the request is authentic, the authentication service may provide authentication proof to the data service frontend. The authentication proof may be an electronic token or other information that is usable by another service, such as the cryptography service, to independently determine that an authentic request was received. In one illustrative example, the PUT request is transmitted with a signature for the PUT request. The PUT request and its signature are provided through the authentication service, which independently computes what a signature should be if authentic. If the signature generated by the authentication service matches that signature provided by the user, the authentication service may determine that the PUT request was authentic and may provide authentication proof in response. Determining whether the PUT request is authentic may also include one or more operations in connection with the enforcement of policy. For example, if the signature is valid but policy otherwise indicates that the PUT request should not be completed (e.g. the request was submitted during a time disallowed by policy), the authentication service may provide information indicating that the request is not authentic. (It should be noted, however, that

such policy enforcement may be performed by other components of the environment 300.) The authentication service may generate the signature, such as by using a key shared by the authentication service and the user. The authentication proof, as noted, may be information from which another service, such as the cryptography service, can independently verify that a request is authentic. For example, using the example of the cryptography service illustrated in FIG. 3, the authentication proof may be generated based at least in part on a key shared by both the authentication service and the cryptography service, such as a key that is inaccessible to other services.

[0039] As illustrated in FIG. 3, the data service frontend, upon receipt of authentication proof from the authentication service, provides plaintext and authentication proof to the cryptography service. The plaintext and authentication proof may be provided according to an API call or other electronic request to the cryptography service (e.g., an Encrypt API call). The cryptography service may analyze the authentication proof to determine whether to encrypt the plaintext.

[0040] It should be noted that additional information may be provided to the cryptography service. For example, an identifier of a key to be used to encrypt the plaintext may be provided as an input parameter to the API call from the data service frontend (which, in turn, may have received the identifier from the user). However, it should be noted that an identifier may not be transmitted to the cryptography service. For instance, in various embodiments, it may be otherwise determinable which key to use to encrypt the plaintext. For example, information transmitted from the data service frontend to the cryptography service may include information associated with the user, such as an identifier of the user and/or an organization associated with the user, such as an identifier of a customer on behalf of which the user has submitted the PUT request. Such information may be used by the cryptography service to determine a default key to be used. In other words, the key may be implicitly specified by information that is usable to determine the key. Generally, determination of the key to be used may be performed in any suitable manner. Further, in some embodiments, the cryptography service may generate or select a key and provide an identifier of the generated or selected key to be used later. Another example API parameter can be an identifier for a master key for the customer account the encryption operation is being performed for.

[0041] As illustrated in FIG. 3, if the authentication proof is sufficient to the cryptography service for the plaintext to be encrypted, the cryptography service can perform one or more cryptographic operations. In an embodiment, the one or more cryptographic operations can

include an operation to generate an envelope key to be used to encrypt the plaintext. The envelope key can be a randomly generated symmetric key or a private key of a key pair. After the envelope key is generated, the cryptographic service can encrypt the envelope key with the master key specified in the API call and cause the encrypted envelope key to be persistently stored (e.g., by storing the encrypted key in a storage service or some other durable storage) or discarded. In addition, the cryptographic service can send a plaintext version of the envelope key as well as and the encrypted envelope key to the data service frontend. The data service can then use the plaintext version of the envelope key to encrypt the plaintext (i.e., the data associated with the encryption request) and cause the envelope key to be stored in persistent storage in association with an identifier for the master key used to encrypt the envelope key. In addition, the data service can discard the plaintext version of the envelope key. As such, in an embodiment after the data service discards the plaintext version of the envelope key it will no longer be able to decrypt the ciphertext.

[0042] In an alternative embodiment, the cryptographic operation can involve encrypting the plaintext. For example, the cryptographic service encrypts the plaintext and provides ciphertext to the data service frontend storage system. The data service frontend may then provide the ciphertext to the data service backend storage system for persistent storage in accordance with its operation. Other information may also be transmitted from the data service frontend to the data service backend storage system. For example, an identifier of the key used to encrypt the plaintext to generate ciphertext may be provided with the ciphertext for storage by the data service backend storage system. Other information, such as metadata identifying the user and/or the user's organization, may also be provided.

[0043] As with all environments described herein, numerous variations are considered as being within the scope of the present disclosure. For example, the flow of information among the various components of the environment 300 may vary from that which is shown. For example, information flowing from one component to another component through an intermediate component (e.g. data from the authentication service to the cryptography service and/or data from the cryptography service to the data service backend storage system) may be provided directly to its destination and/or through other intermediate components of the environment 300 (which are not necessarily included in the figure). As another example, PUT requests (and, below, GET requests) are provided for the purpose of illustration. However, any suitable request for performing the operations described may be used.

[0044] FIG. 4 shows an illustrative example of a process 400 which may be used to store data in a data storage service in accordance with an embodiment. The process 400 may be

performed, for example, by the data service frontend illustrated in FIG. 3. Some or all of the process 400 (or any other processes described herein, or variations and/or combinations thereof) may be performed under the control of one or more computer systems configured with executable instructions and may be implemented as code (e.g., executable instructions, one or more computer programs or one or more applications) executing collectively on one or more processors, by hardware or combinations thereof. The code may be stored on a computer-readable storage medium, for example, in the form of a computer program comprising a plurality of instructions executable by one or more processors. The computer-readable storage medium may be non-transitory.

[0045] As illustrated in FIG. 4, the process 400 includes receiving 402 a PUT request. The PUT request may be electronically received over a network and may include information associated with the request, such as information required for authentication, such as an electronic signature of the PUT request. In response to having received the PUT request, the process 400 may include submitting 404 an authentication request. For example, the system performed in the process 400 may submit (e.g., via an appropriately configured API call) an authentication request to a separate authentication service, such as described above in connection with FIG. 3. Similarly, a data service frontend that performs its own authentication may submit the authentication request to an authentication module implemented by the data service frontend. Generally, the authentication request may be submitted in any suitable manner in accordance with the various embodiments.

[0046] Upon submission of the authentication request, an authentication response is received 406 by the entity to which the authentication request was submitted 404. For example, referring to FIG. 3, the authentication service may provide a response to the data service frontend that includes proof of the authentication for use by other services. Other information, such as an indication of whether or not authentication was successful, may also be transmitted. A determination may be made 408 whether the request is authentic. Authenticity of the request may depend from one or more factors checked by an entity, such as by an authentication service, or a combination of entities that collectively perform such checks. Authenticity may, for example, require that the request provide necessary valid credentials (e.g., an electronic signature generated by a secret key shared by the checking entity) and/or that policy allows the request to be fulfilled. From the perspective of a system that submits 404 an authentication request and receives an authentication response, authenticity may depend from the received authentication response. Accordingly, in an embodiment, the determination 408 whether the request is authentic may be performed based

at least in part of the received authentication response. For example, if authentication was not authentic, the authentication response so indicates and the determination 408 may be made accordingly. Similarly, the response may implicitly indicate that the authentication request is authentic, such as by not including information that would be included if the request was authentic. If it is determined 408 that the PUT request is not authentic, then the PUT request may be denied 410. Denying the PUT request may be performed in any suitable manner and may depend upon the various embodiments in which the process 400 is being performed. For example, denying 410, the PUT request may include transmitting a message to a user that submitted the PUT request. The message may indicate that the request was denied. Denying the request may also include providing information about why the request was denied, such as an electronic signature not being correct or other reasons that may be used for determining how to resolve any issues that resulted in the PUT request not being authentic or authorized.

[0047] If it is determined 408 that the PUT request is authentic and authorized, then, in an embodiment, the process 400 includes performing 412 one or more cryptographic operations that result in the plaintext being encrypted. For example, a request (e.g., an appropriately configured API call) may be submitted to a cryptography service to provide a key to be used for performing the one or more cryptographic operations. The request provided to the cryptography service may be provided with proof of the PUT request being authentic so that the cryptography service can independently determine whether to perform the cryptographic operation (e.g., to encrypt the plaintext and provide ciphertext or generate an envelope key that can be used to encrypt the plaintext). However, in various embodiments, authentication proof may not be provided to the cryptography service and, for example, the cryptography service may operate in accordance with the request that it receives. For example, if the cryptography service receives a request from the data service frontend, the cryptography service may rely on the fact that the data service frontend has already independently verified authentication of the request. In such an embodiment and other embodiments, the data service frontend may authenticate itself with the cryptography service to provide an additional layer of security. The cryptography service may generate or otherwise obtain a key, encrypt the obtained key or otherwise obtain the encrypted key (e.g., from memory), and provide the obtained key and the encrypted obtained key in response to the request. The obtained key may be encrypted using a key identified in the request to the cryptography service. The obtained key may be used to encrypt the plaintext and, after encrypting the plaintext, the obtained key may be discarded (e.g., irrevocably removed from memory). In alternate embodiment, a system performing the process 400 may generate or otherwise obtain

the key used to perform the one or more cryptographic operations, provide the obtained key to the cryptography service to be encrypted.

[0048] In some embodiments, performing the one or more cryptographic operations may result in ciphertext being generated. Ciphertext generated as a result of one or more cryptographic operations may be stored 414 for possible retrieval at a later time. As noted above, storage of the ciphertext may include storage of additional information that would enable the decryption of the ciphertext at later time. For instance, the ciphertext may be stored with an identifier of a key used to encrypt the plaintext into the ciphertext so that the key having that identifier may later be used to decrypt the ciphertext to obtain the plaintext. Storage of the ciphertext may also be performed in any suitable manner. For example, storage of the ciphertext may be performed by a data service backend storage system, such as described above.

[0049] FIG. 5, accordingly, shows an illustrative example of an environment 500 and the flow of information illustrating how plaintext may be obtained. The environment 500 in this example includes an authentication service, a cryptography service, a data service frontend and a data service backend storage system. The authentication service, cryptography service, data service frontend and a data service backend storage system may be systems such as described above. As illustrated in FIG. 5, the data service frontend is configured to receive a GET request from a user and provide plaintext in response. In order to do this, the data service frontend may also be configured to submit authentication requests to an authentication service which itself may be configured to, if appropriate, provide to the data service frontend authentication proof. The data service frontend may also be configured to send a request to the cryptographic service to cause it to execute one or more cryptographic operations related to decrypting the data. In an embodiment where envelope keys are used, the data service can submit a request (e.g., an API call) to the cryptographic service that includes or specifies the encrypted envelope key (or an identifier for the encrypted envelope key) authentication proof, and an identifier of the master key used to encrypt the envelope key to the cryptographic service. The cryptographic service can determine whether the authentication proof is sufficient to allow the operation and, if the authentication proof is sufficient, decrypt the envelope key. The decrypted envelope key can be sent back to the data service, which can use the key to decrypt the encrypted plaintext. The data service can then discard the decrypted plaintext key.

[0050] In an alternative embodiment, the data service frontend may be configured to provide the received authentication proof to the cryptography service with ciphertext for the

cryptography service to decrypt. The cryptography service may, accordingly, be configured to determine whether the authentication proof is sufficient to allow decryption of the ciphertext and, if the authentication proof is sufficient, decrypt the ciphertext using an appropriate key (which may be identified to the cryptography service by the data service frontend), and provide the decrypted ciphertext (plaintext) to the data service frontend. To provide ciphertext to the cryptography service, the data service frontend may be configured to obtain (e.g., via an appropriately configured API call) the ciphertext from the data service backend storage system.

[0051] FIG. 6 shows an illustrative example of a process 600 which may be used to obtain plaintext in accordance with various embodiments. The process 600 may be performed, for example, by the data service frontend system (data service frontend) illustrated above in connection FIG. 5, although the process 600 and variations thereof may be performed by any suitable system. In an embodiment, the process 600 includes receiving 602 a GET request (or other appropriate request) from a user. Receiving the GET request may be performed such as described above in connection with other types of requests. Upon receipt 602 of the GET request, an authentication request may be submitted 604 to an authentication service or in any manner such as described above. An authentication response may, accordingly, be received. Based at least in part on the receive authentication response, a determination may be made 608 whether the GET request is authentic. If it is determined 608 that the GET request is not authentic, the process 600 may include denying 610 the request which, as described above, may be performed in various manners in accordance with the various embodiments.

[0052] If it is determined 608 that the GET request is authentic, the process 600 may include retrieving ciphertext from storage. Retrieving 612 ciphertext from storage may be performed in any suitable manner. For example, referring to the environment 500 discussed above in connection with FIG. 5, the data service frontend may submit a request for the ciphertext to the data service backend storage system and may receive the ciphertext in response. Generally, ciphertext may be obtained from storage in any suitable manner. Upon receipt of the ciphertext, the process 600 may include performing 614 one or more operations relating to decrypting the ciphertext. For example, in an embodiment, the data storage service may send a request to the cryptographic service to perform one or more cryptographic operations relating to decrypting the ciphertext 614. In one example configuration, the data service can send the cryptographic service an API call that includes the encrypted envelope key (or an identifier for the encrypted envelope key) authentication proof, and an identifier of

the master key used to encrypt the envelope key to the cryptographic service. The cryptographic service can determine whether the authentication proof is sufficient to allow the operation and, if the authentication proof is sufficient, decrypt the envelope key. The decrypted envelope key can be sent back to the data service, which can use the key to decrypt the encrypted plaintext.

[0053] In another configuration, the ciphertext may be provided to a cryptography service such as the cryptography service described above in connection with FIG. 5. Other information may also be provided to the cryptography service such as proof of authentication that can be used by the cryptography service to determine whether or not to decrypt the ciphertext. In addition, in some embodiments, an identifier of a key to be used by the cryptography service to decrypt the ciphertext may be provided to the cryptography service. However, in other embodiments, the key may be implicitly indicated to the cryptography service. For example, the cryptography service may use a default key associated with a customer that is indicated to the cryptography service. Generally, any manner by which the cryptography service can determine which key to use to decrypt the ciphertext may be used.

[0054] As illustrated in FIG. 6, after the ciphertext is decrypted, the process 600 may include providing 616 a response to the GET request. Providing a response to the GET request may be performed in various ways in accordance with the various embodiments. For example, providing the response to the GET request may include providing the plaintext. In other embodiments, the plaintext may be a key that is used to decrypt other encrypted information which is then provided in response to the GET request. Generally, depending on the role of the plaintext in a particular embodiment of the disclosure, providing a response to the GET request may be performed in various ways.

[0055] As noted, various embodiments of the present disclosure allow for data to be stored by a data storage service in various ways. FIG. 7 shows an illustrative example of an environment 700 with arrows indicating information flow in accordance with such embodiment. As illustrated in FIG. 7, environment 700 includes an authentication service, a cryptography service, a data service frontend and a data service backend storage system, such as described above. In this particular example, the data service frontend is a computer system configured to receive PUT requests from various users. PUT requests may include or specify data objects to be stored by a data service backend storage system. PUT requests may also specify a key identifier for a key to be used to encrypt the data object. The data service frontend may also be configured to interact with an authentication service, such as described above, in order to provide authentication proof to a cryptography service that is operable to

receive keys and key identifiers and provide in response keys encrypted by the keys identified by the key identifiers. The data service frontend may then cause storage in a data service backend storage system. The data that may be stored may include a data object encrypted by the key. The data that may be stored may also include the key encrypted by the key identified by the key identifier. As discussed elsewhere, herein, the encrypted data object and encrypted key may be stored in different services.

[0056] As illustrated in FIG. 7, the data service frontend is configured to provide encrypted information to the data service backend storage system for storage. In this example, the data service frontend is configured to provide a data object encrypted under a key and the key encrypted under another key having a KeyID. It should be noted that, for the purpose of illustration, curly bracket notation is used to denote encryption. In particular, information inside of curly brackets is the information that is encrypted under a key specified in subscript. For example, $\{\text{Data Object}\}_{\text{Key}}$ denotes that the data "Data Object" is encrypted under the key "Key." It should be noted that a key identifier may also appear in subscript using this curly bracket notation. When a key identifier appears in subscript, the information inside the curly brackets is encrypted under a key identified by the key identifier. For example, $\{\text{Data Object}\}_{\text{KeyID}}$ denotes that the data object "Data Object" is encrypted under a key identified by the key identifier "KeyID." Similarly, $\{\text{Key}\}_{\text{KeyID}}$ denotes that the key "Key" is encrypted under the key identified by the key identifier "KeyID." In other words, this disclosure makes use of both keys and key identifiers in subscript and the meaning of the subscript should be clear from context. The ciphertext may include additional metadata usable to determine the identity of the associated decryption key.

[0057] FIG. 8 shows an illustrative example of a process 800 which may be performed to store a data object in a data storage system, such as the data service backend storage system described above in connection with FIG. 7. The process 800 may be performed by any suitable system, such as by the data service frontend system described above in connection with FIG. 7. In an embodiment, the process 800 includes receiving 802 a PUT request for a data object. Receiving the PUT request for the data object may be performed in any suitable manner, such as described above. It should be noted that the data object can be received in connection with the request or may be received from another service. For example, the request may include an identifier for a data object that may be obtained from another service using the identifier. As with other processes described above, the process 800 in an embodiment includes submitting 804 an authentication request and receiving 806 an authentication response. The authentication response that was received 806 may be used to

determine 808 whether the PUT request is an authentic request. If it is determined 808 that the PUT request is not authentic, the process 800 may include denying 810 the request such as described above. If it is determined 808 that the PUT request is authentic, the process 800 may include obtaining 812 a key identifier (KeyID), such as a keyID for a master key used to encrypt envelope keys. Obtaining 812 the KeyID may be performed in any suitable manner and the manner in which the KeyID is obtained may vary in accordance with the various embodiments. For example, as illustrated in FIG. 7, the PUT request may specify the KeyID. As another example, the identity of the user, or otherwise associated with the user, may be used to obtain an identifier or a default key. As another example, the ciphertext may provide an indication of the associated key ID. As yet another example, one or more policy determinations may be used to determine which key identifier to obtain.

[0058] In an embodiment, the process 800 also includes generating 814 a key, such as an envelope key. Generating the key may be performed in any suitable manner by, for example, the cryptographic service or the service requesting encryption operations from the cryptographic service (e.g., a data storage service). For example, the key may be generated using a key derivation function using appropriate input to the key derivation function. Example key derivation functions include KDF1, defined in IEEE Std 1363 2000, key derivation functions defined in ANSI X9.42, and HMAC-based key derivation functions, such as the HMAC-Based Extract-and-Expand Key Derivation Function (HKDF) specified in RFC 5869. As another example, the key may be generated by a random or pseudo random number generator, a hardware entropy source or a deterministic random bit generator such as is specified by National Institute of Standards and Technology Special Publication (NIST SP) 800-90A. It should be noted that while FIG. 8 shows the process 800 including generating 814 a key, the key may be obtained in other ways such as by retrieval from storage. In other words, the key may have been pre-generated.

[0059] Continuing with the process 800 illustrated in FIG. 8, in an embodiment, the process 800 includes using 816 the generated key to encrypt a data object. For example, in an embodiment where the cryptographic service generates the key, the cryptographic service can provide the key, the KeyID, and an encrypted copy of the key to the data service. For example, referring to FIG. 7, the data service frontend may receive the envelope key and the KeyID for the master key used to encrypt the envelope key from the cryptography service with any other relevant information, such as authentication proof. The plaintext copy of the encryption key may then be used to encrypt the data object. The plaintext copy of the encryption key can be discarded and the encrypted data object as well as the encrypted key

may then be stored 818. For example, referring to FIG. 7, the data service frontend may transmit to the encrypted data object and the encrypted key to the data service backend storage system for storage. In a configuration where the service generates the key, the service can provide the key and the KeyID to the cryptography service. For example, the data service frontend may send the envelope key and the KeyID for the master key used to encrypt the envelope key to the cryptography service with any other relevant information, such as authentication proof. The plaintext copy of the encryption key may then be used to encrypt the data object. The service can discard the plaintext copy of the encryption key and the encrypted data object as well as the encrypted key may then be stored. For example, referring to FIG. 7, the data service frontend may transmit to the encrypted data object and the encrypted key to the data service backend storage system for storage.

[0060] The encrypted data object and the encrypted envelope key may be stored without the plaintext version of key, that is, the plaintext key may be inaccessible to the data service backend storage system and one or more other systems. The key under which the data object is encrypted (e.g., the master key) may be made inaccessible in any suitable manner. In some embodiments this is achieved by storing it in memory accessible only to the cryptographic service. In some other embodiments this can be achieved by storing the master key in a hardware or other security module or otherwise under the protection of a hardware or other security module. In some embodiments, the memory location storing the plaintext envelope key (e.g., memory of the data service) may be allowed to be overwritten or memory location storing the key may be intentionally overwritten to render inaccessible the key to the data service frontend. As another example, the plaintext envelope key may be maintained in volatile memory that eventually ceases to store the key. In this manner, the envelope key is only accessible if it is decrypted using a key identified by the KeyID or otherwise obtained in an unauthorized manner, such as by cracking the key without the key identified by the KeyID, which may be computationally impractical. In other words, the key identified by the KeyID is required for authorized access to the key under which the data object is encrypted. Thus, if the data service backend storage system of FIG. 7 is compromised, such compromise would not provide access to the unencrypted data object because decrypting the data object would require access to the key, which is only obtainable through decryption using the key identified by the KeyID or through other ways which are not computationally feasible.

[0061] As noted, various embodiments of the present disclosure allow users to both store data objects and retrieve them in a secure manner. FIG. 9, accordingly, shows an illustrative example of an environment 900 which may be used to obtain data objects from storage. As

illustrated in FIG. 9, the environment 900 includes an authentication service, a cryptography service, a data service frontend system and a data service backend storage system. The authentication service, cryptography service, data service frontend and data service backend storage system may be computer systems such as described above. As illustrated in FIG. 9, the data service frontend system is configured to receive data object requests and provide data objects in response. In order to provide the data objects in response, the data storage frontend system in this embodiment is configured to interact with the authentication service, the cryptography service, and the data service backend storage system as illustrated in FIG. 9. For example, in various embodiments, the data service frontend system is configured to submit authentication requests to the authentication service and receives authentication proof in response to the requests. As another example, the data service frontend is configured to provide keys encrypted by a key identified by a KeyID and authentication proof to a cryptography service which is operable to determine whether to provide the key based, at least in part, on the authentication proof and, if determined to provide the key, then provide the key to the data service frontend. The data service frontend may also be configured to provide other information, such as the KeyID, to the cryptography service. Although, in some embodiments, the KeyID may be implicitly indicated to the cryptography service, such as through association with other information provided to the cryptography service. It should also be noted that, in some embodiments, the user provides the KeyID to the data service frontend in connection with submitting requests to the data service frontend. Also, as illustrated in FIG. 9, the data service frontend in an embodiment is configured to request the data object from the data service backend storage system and receive in response the data object encrypted by the key and the key encrypted by a key identified by the KeyID. In some embodiments the cryptographic service may be operable to refuse to perform decryption of a ciphertext not generated using a key associated with a specified KeyID.

[0062] The data service frontend, in an embodiment, is configured to use the key received from the cryptography service to decrypt the data object and provide the decrypted data object to the user. FIG. 10, accordingly, shows an illustrative example of a process 1000 which may be used to provide the decrypted object in accordance with various embodiments. The process 1000 may be performed by any suitable system such as the data service frontend system described in connection with FIG. 9. In an embodiment, the process 1000 includes receiving 1002 a GET request for a data object. Receiving the GET request for the data object may be performed in any suitable manner such as described above in connection with other types of requests. For example, the GET request for the data object may include information used to authenticate the request and/or other information. The process 1000,

accordingly, in an embodiment, as with other processes described herein, includes submitting 1004 an authentication request to an authentication system and receiving 1006 an authentication response. Submitting the authentication request and receiving the authentication response may be performed in any suitable manner, such as described above. The authentication response may be used to determine 1008 whether or not the GET request is authentic. If it is determined 1008 that the GET request is not authentic, the process 1000 in an embodiment includes denying 1010 the request. If, however, it is determined 1008 that the GET request is authentic, the process 1000 in an embodiment includes retrieving 1012 the encrypted data object and an encrypted key from storage. For example, the data service frontend system may obtain the encrypted data object and encrypted key from the data service backend storage system illustrated above in connection with FIG. 9.

[0063] In an embodiment, the process 1000 includes providing 1014 the encrypted envelope key to a cryptography service. Providing 1014 the encrypted envelope key to the cryptography service may be performed in any suitable manner and may be provided with other information, such as authentication proof that enables the cryptography service to determine whether to decrypt the encrypted key. In addition, providing 1014 the encrypted envelope key to the cryptography service may include providing an identifier of a key required for authorized decryption of the encrypted envelope key to enable the cryptography service to select a key identified by the identifier from among multiple keys managed by the cryptography service. As noted above, however, keys may be implicitly identified. The cryptography service may, accordingly, select an appropriate key and decrypt the encrypted key. Accordingly, in an embodiment, the process 1000 includes receiving 1016 the decrypted envelope key from the cryptography service. For example, if the cryptography service determines that the authentication proof is valid and/or that decryption of the encrypted is allowable according to any applicable policies, the cryptography service may provide the decrypted key to a system attempting to decrypt the data object. The data object may then be decrypted 1018 using the decrypted envelope key. The decrypted data object may then be provided 1020 to the requestor, such as the user or other system that submitted the GET request.

[0064] In many instances, it is desirable for users (i.e., generally devices utilizing a cryptography service) to interact directly with the cryptography service. FIG. 11 accordingly shows an illustrative example of an environment 1100 which allows for direct user access to a cryptography service. In environment 1100, included is an authentication service, a data service frontend and a data service backend storage system. The authentication service, data

service frontend and data service backend storage system may be as described above. For example, the data service frontend may be configured to receive and respond to requests from users as illustrated in FIG. 11 over a suitable network. As part of responding to requests from the users over the network, the data service frontend may also be configured to interact with the authentication service in order to determine whether user requests are authentic and/or enforce policy on the requests. The data service frontend may also be configured to interact with the data service backend storage system as part of fulfilling user requests. User requests may include, for example, PUT requests to store data in the backend storage system and GET requests to retrieve data from the data service backend storage system. As above, other requests may also be used in accordance with the various embodiments, such as requests to delete data stored in the data service backend storage system, requests to update the data stored in the data service backend storage system and the like.

[0065] In the particular example of FIG. 11, in the environment 1100, the cryptography service includes a cryptography service frontend and a data service backend. As with the data service frontend, the cryptography service frontend is configured to receive and respond to requests from users over the network. The cryptography service frontend is also configured to interact with the authentication service to determine whether user requests are authentic. Determining whether user requests are authentic may be performed in a simple manner such as described above. It should be noted that, while the cryptography service frontend and the data service frontend interact with the same authentication service, the cryptography service frontend and the data service frontend may interact with different authentication services. Further, the cryptography service frontend may be configured to enforce policy when responding to user requests.

[0066] The cryptography service frontend, in an embodiment, is configured to interact with the cryptography service backend. The cryptography service backend is configured, in accordance with instructions received from the cryptography service frontend, to perform cryptographic operations. Cryptographic operations include encryption, decryption and hash calculations and others. The environment 1100 may be used, for example, by users to have plaintext encrypted by the cryptography service so that the encrypted data can be stored in the data service backend storage system. Examples of such use of the environment 1100 are provided below. In addition, example details of an example cryptography service are also provided below.

[0067] Data may be stored in the data service backend storage system in any suitable manner such as described above. For example, techniques for storing encrypted data in the

backend storage system described above may be used in the environment 1100. For example, while not illustrated, the data service frontend may communicate with the cryptography service frontend to cause the cryptography service backend to encrypt data which may then be stored in the data service backend storage system. The encrypted data may be a data object and/or an encrypted key that was used to encrypt a data object. In the environment 1100, data may be placed into the data service backend storage system in other ways as well. For example, users may provide plaintext to be encrypted by the cryptography service and may receive ciphertext in response. The users may then interact or may submit a request to the data service frontend to request that the ciphertext be stored in the data service backend storage system. The data service frontend, in this example, may store the ciphertext in any manner. For instance, the data service frontend and backend storage systems may be configured to be indifferent to whether data is encrypted.

[0068] In addition, as with all environments illustrated herein, an additional frontend system may be logically located between the users and the data service frontend and the cryptography service frontend and possibly other frontend systems in order to coordinate actions between the systems. For example, in some embodiments, users may interact with a frontend system that itself interacts with the cryptography service frontend and data service frontend so that operations from the perspective of the user are simpler. For example, a user may request that a data object be encrypted and stored and a frontend system responds to the request by appropriate interactions with the cryptography service frontend and data service frontend. From the perspective of the user, however, such may be performed by a single request. Other variations are also within the scope of the present disclosure.

[0069] FIG. 12 shows an illustrative example of an environment 1200 which may be used to implement various embodiments of the present disclosure. In FIG. 12, the environment 1200 is configured to enable users to store ciphertext in a data service backend storage system. As illustrated in FIG. 12, accordingly, the environment 1200 includes a data service frontend, a data service backend storage system, an authentication service, a cryptography service frontend and a cryptography service backend. The data service backend storage system, the data service frontend, the authentication service, the cryptography service frontend and the cryptography service backend may be systems such as described above in connection with FIG. 11. For example, as illustrated in FIG. 12, the data service frontend is configured to receive and respond to user requests and may also be configured to enforce policy on user requests. The data service frontend, as part of responding to requests, may be configured to submit authentication requests to the authentication service and receive

authentication proof in response in response. Upon successful authentication, the data service frontend may be further configured to interact with the data service backend storage system to obtain encrypted data objects and possibly unencrypted data objects from the data service backend storage system, which may be then provided to a user.

[0070] As illustrated in FIG. 12, the cryptography service frontend is also configured to submit authentication requests to the authentication service and receive, in response, authentication proof. Authentication proof may be used to obtain services from the cryptography service backend. For example, the cryptography service frontend may be configured to provide ciphertext to the cryptography service backend with authentication proof and the cryptography service backend may be configured to decrypt the ciphertext and provide the ciphertext in return. As illustrated in FIG. 12, the ciphertext may be an encrypted key and the cryptography service backend may decrypt the encrypted key and provide the decrypted key, that is a plaintext key, to the cryptography service frontend, which is further configured to provide the plaintext key to the user. The user may then use the key to decrypt the encrypted data object received from the data service frontend or to decrypt encrypted data objects stored within the user's domain (e.g., within a user operated or controlled data center or computer system). In this example, the user may have obtained the encrypted key from the data service frontend. For example, the user may have submitted a request to the data service frontend for the data object and/or the key used to encrypt the data object. While illustrated in FIG. 11 as a single request, the separate requests may be made for both the data object and the key. As illustrated in FIG. 11, the data service frontend may obtain the encrypted data object and encrypted key from the data service backend storage system and provide the encrypted data object and encrypted key to the user.

[0071] It should be noted that, as with all environments illustrated herein, variations are considered as being within the scope of the present disclosure. For example, FIG. 12 shows a data object encrypted under a key and the key encrypted by another key identified by a key identifier being provided to the user. Further levels of encryption may also be used. For example, the data object may be encrypted under a key that is only accessible to the user (and/or that is not accessible by the other components of the environment 1200). A key used to encrypt the data object may also be encrypted under the key that is only accessible to the user. In this example, unauthorized access to the components of the environment 1200 (absent the user) still does not provide access to the unencrypted contents of the data object since access to the user's key is still required for authorized decryption.

[0072] As another example, in the environment 1200 illustrated in FIG. 12, the data service frontend and the data service backend storage system do not have access to the plaintext data stored by the data service backend storage system because the data service frontend and the data service backend storage system do not have access to a key needed to decrypt the encrypted data. In some embodiments, however, access may be granted to the data service frontend and/or the data service backend storage system. For instance, in an embodiment, temporary access to the key may be provided to the data service frontend to enable the data service frontend to obtain encrypted data, decrypt the encrypted data, use the decrypted data for a particular purpose (e.g., indexing), and then delete or otherwise lose access to the decrypted data. Such actions may be governed by policy enforced by the data service frontend and/or cryptography service and may require authorization from the user.

[0073] FIG. 13 shows an illustrative example of a process 1300 which may be used to obtain an encrypted data object and an encrypted key such as from a data service backend storage system such as described above. The process 1300, for example, may be performed by the data service frontend system described above in connection with FIG. 12. In an embodiment, the process 1300 includes receiving 1302 a GET request for an encrypted data object. Receiving the GET request may be performed in any suitable manner such as by receiving the request via an API call to the data service frontend system. As a result of having received the GET request, process 1300 may include submitting 1304 an authentication request and receiving 1306 an authentication response. Submitting 1304 the authentication request and receiving 1306 the authentication response may be performed in any suitable manner such as described above. The authentication response may be used to determine 1308 whether the GET request is authentic. If it is determined 1308 that the GET request is not authentic, the process 1300 may include denying 1310 the GET request. Denying 1310 the GET request may be performed in any suitable manner such as described above. If, however, it is determined 1308 that that GET request is authentic, the process 1300 may include providing 1312 the encrypted data object with an encrypted key that, when decrypted, is usable to decrypt the encrypted data object. It should be noted that, as with all processes described herein, numerous variations are considered as being within the scope of the present disclosure. For example, the process 1300 may be configured to respond to the GET request, when authentic, by providing the encrypted data object but without providing an encrypted key. A requester, that is a user or system that submitted the GET request, may obtain the encrypted key in other ways. For example, in some embodiments, users may store encrypted keys themselves in a data storage system under the user's control. As another example, one storage service may store the encrypted data object and another service may

store the encrypted key and the user may obtain the encrypted data object and encrypted key from the respective services. As another example, another service or a third party to the user may be used to store encrypted keys and users may obtain encrypted keys upon request. Generally, any way by which encrypted keys may be provided may be used.

[0074] As illustrated in FIG. 13, the process 1300 may result in an entity having been provided a data object and an encrypted key usable to decrypt the data object. In various embodiments, the encrypted key must be decrypted in order to decrypt the data object. FIG. 14, accordingly, shows an illustrative example of a process 1400 which may be used to provide a decrypted key to an entity in need of such a decrypted key in order to use the decrypted key for decryption of an encrypted data object. The process 1400 may be performed by any suitable system such as by the cryptography service frontend system described above in connection with FIG. 12. In an embodiment, the process 1400 includes receiving 1402 a decryption to decrypt a key using another key having a specified KeyID. While the process 1400 is described in connection with decryption of a key, it should be noted that the process 1400 may be adapted for decryption of data in general. The decryption request may be received 1402 in any suitable manner such as described above (e.g., via an appropriately configured API call). Further, the decryption request may be received by any entity appropriate to the context in which the process 1400 is being performed. For instance, the decryption request may originate from the user or from another system, such as the data service frontend discussed above. The decryption request may also include data (e.g. a key) to be decrypted or a reference thereto. The KeyID may be specified also in any suitable manner. For example, in some embodiments, the decryption request includes the KeyID or a reference to the KeyID, that is, information that can be used to determine the KeyID. As discussed above, the KeyID may also be implicitly specified. For example, the KeyID may be obtained through association with available data such as an identity of a requester that submitted the decryption request. For instance, the key corresponding to the KeyID may be a default key for the requestor or for the entity on behalf of which the request was submitted.

[0075] The process 1400, in an embodiment, includes submitting 1404 an authentication request and receiving 1406 an authentication response. Submitting 1404 the authentication request and receiving 1406 the authentication response may be performed in any suitable manner such as described above. Further, as described above, the received authentication response may be used to determine 1408 whether the GET request is authentic. If it is determined 1408 that the GET request is not authentic, the process 1400 may include denying 1410 the GET request. Denying 1410 the GET request may be performed in any suitable

manner such as is described above. If, however, it is determined 1408 that the GET request is authentic, the process 1400 may include accessing policy information for the specified KeyID and/or for the requester. Policy information may include information including one or more policies on the KeyID and/or the requester.

[0076] In an embodiment, the accessed policy information is used to determine 1414 whether any applicable policy allows decryption of the key having the specified KeyID. If it is determined 1414 that the policy does not allow decryption of the key specified by the KeyID, the process 1400 may include denying 1410 the GET request such as described above. If, however, it is determined 1414 that policy allows decryption of the key having the specified KeyID, the process 1400 may include decrypting 1416 the key using the key identified by the KeyID. Once the key has been decrypted, using a key having the KeyID, the decrypted key may then be provided 1418 such as by transmission over a network to the requester that submitted the decryption request (or, in some embodiments, another authorized destination).

[0077] As illustrated in the environment 1200 discussed above, a user may obtain encrypted data objects and keys for decrypting the data objects in various ways. FIG. 15 shows an illustrative example of a process 1500 which may be used to obtain plaintext in accordance with various embodiments. The process 1500 may be performed by any suitable system such as by a system being operated and/or hosted by a user such as described in connection with FIG. 12. Other suitable systems include systems operating on behalf of users and not necessarily according to real time user input provided but perhaps according to preprogrammed processes.

[0078] In an embodiment, the process 1500 includes receiving 1502 ciphertext from a data storage service. Requesting 1502 ciphertext from a data storage service may be performed in any suitable manner such as described above. For example, a system performing the process 1500 may request 1502 ciphertext, using an appropriately configured API call in the environment 1200 illustrated above in connection with FIG. 12 and/or by the process 1300 described above in connection with FIG. 13.

[0079] The process 1500 may also include receiving ciphertext and an encrypted key. Receiving ciphertext and encrypted key may be performed in any suitable manner. For example, the ciphertext and encrypted key may be received in a response to the request for the ciphertext from a data storage service. Generally, however, the ciphertext and encrypted key may be received 1504 in other suitable ways. For example, the request to receive

ciphertext from the data storage service may be an asynchronous request and ciphertext may be received 1504 pursuant to another request that is subsequently submitted. Further, the ciphertext and encrypted key may be provided in a single response or may be obtained separately such as by different responses (which may be from the same or from different systems). As another example, a system performing a process 1500 may locally or otherwise store encrypted keys and the encrypted key may be received from local memory.

[0080] In an embodiment, the process 1500 includes requesting decryption of the encrypted key, using a key having a specified KeyID. The KeyID may be specified in any suitable manner such as described above. Further, it should be noted that the system performing the process 1500 may be able to specify the KeyID in any suitable manner. For example, the encrypted key and/or information provided therewith may specify the KeyID. As another example, the system performing the process 1500 may have local or remote access to information that enables determining the KeyID. A local or remote database, for instance, may associate data object identifiers with key identifiers for keys used to encrypt the data objects. Generally, any manner in which the system can be enabled to specify the KeyID may be used. Further, in some embodiments, the KeyID need not be specified, such as when information provided to a cryptography service is sufficient to determine the KeyID. The request 1506 for decryption of the encrypted key may be performed in any suitable manner such as in connection with an environment discussed above in connection with FIG. 12 and/or by performance of the process 1400 described above in connection with FIG. 14.

[0081] The process 1500, in an embodiment, includes receiving 1508 the decrypted key. Receiving 1508 the decrypted key may be performed in any suitable manner. For example, the decrypted key may be received in response to a request for decryption of the encrypted key. As another example, the request for decryption of the encrypted key may be an asynchronous request and another request may have been submitted for receiving the decrypted key. Generally, the decrypted key may be received in any suitable manner. Further, as with all information flowing from one device to another, the passage of information may be performed using secure channels. For instance, the decrypted key may be once again encrypted for decryption by an entity receiving the decrypted key. Generally, any manner of secure communication may be used to pass information from one entity to another.

[0082] Once the decrypted key has been received 1508, the process 1500 may include using 1510 the decrypted key to decrypt 1510 ciphertext and therefore obtain plaintext. It should be noted that, as with all processes described herein, variations are considered as

being within the scope of the present disclosure. For example, the process 1500 shows a request for ciphertext and a request for decryption of an encrypted key being performed sequentially. However, as with many operations described herein in connection with the various processes, operations need not be performed sequentially in various embodiments. For example, if a system performing the process 1500 has access to the encrypted key prior to requesting the ciphertext, or is otherwise able to do so, the system may request ciphertext and request decryption of the encrypted key in parallel or in an order different from that which is illustrated. Other variations are also considered as being within the scope of the present disclosure.

[0083] As discussed above, various embodiments of the present disclosure are directed to providing cryptography services. Cryptography services may be provided by a cryptography service system such as described above. FIG. 16 accordingly shows an illustrative example of a cryptography service 1600 in accordance with various embodiments. As illustrated in FIG. 16 and as discussed above, the cryptography service 1600 is logically comprised of a frontend system and a backend system. Both the frontend system and the backend system may be implemented by one or more computer systems configured to perform operations described herein. For example, as illustrated in FIG. 16, the frontend system of the cryptography service 1600 implements a request API and a policy configuration API. The request API, in an embodiment, is an API configured for requesting cryptographic and other operations to be performed by the cryptography service. Thus, requests may be made to the frontend system via the request API in order for such cryptographic operations to be performed by the cryptography service.

[0084] The request API may be configured with the following example, high-level, requests available:

CreateKey(KeyID)

Encrypt(KeyID, Data, [AAD])

Decrypt(KeyID, Ciphertext, [AAD])

Shred(KeyID)

ReKey(Ciphertext, OldKeyID, NewKeyID).

[0085] A CreateKey(KeyID) request, in an embodiment, causes the cryptography service to create a key identified by the KeyID identified in the request. Upon receipt of a request, the cryptography service may generate a key and associate the key with the KeyID. It should be

known that KeyID's may be, but are not necessarily unique identifiers. For instance, a KeyID may identify a family of keys. For example, in some embodiments, key rotation is performed. Key rotation may involve replacing keys with other keys to prevent collection of enough decrypted data to allow practical cracking of a cipher used. If performed at the direction of an entity different from the cryptography service, use of the CreateKey(KeyID) request may cause the cryptography service to create a new key to replace an old key identified by the KeyID. The old key may remain identified by the KeyID, but may, for instance, be only used for decryption (of data that has already been encrypted using the old key) and not for future encryption. As another example, in some embodiments, users of the cryptography service provide their own key identifiers and there is a possibility that two different customers may provide the same identifier. In such instances, the identifier may not uniquely identify a key or even uniquely identify a family of keys. Various measures may be in place to address this. For example, an identity or other information associated with a user of the cryptography service may be used to identify the proper key or family of keys. In still other embodiments the cryptographic service may assign a KeyID randomly, sequentially, or using any other method.

[0086] It should be noted that, when a KeyID does not uniquely identify a key, various systems may be in place to enable proper functionality. For example, in various embodiments, a family of keys identified by a KeyID is finite. If a decryption operation using a key identified by a KeyID is requested, additional data (e.g., a time stamp of when the encryption was performed) may enable determining the proper key to use. In some embodiments, ciphertexts may include information indicating a key version. In some embodiments, all possible keys are used to provide different decryptions of the data. Since there are a finite number of keys, the proper decryption may be selected from those provided. In some embodiments, decryption with a key is performed in a manner that enables the cryptographic service to detect that the ciphertext was not generated based at least in part on the key, such as by using authenticated encryption. Other variations are also considered as being within the scope of the present disclosure.

[0087] An Encrypt(KeyID, Data, [AAD]) request may be used to cause the cryptography service to encrypt the specified data using a key identified by the KeyID. Additional Authenticated Data (AAD) may be used for various purposes and may be data that is not necessarily encrypted, but that is authenticated, e.g. by an electronic signature, a message authentication code or, generally, a keyed hash value included with the AAD. In some embodiments, the ciphertext is generated including at least a portion of the AAD. In some

other embodiments the AAD is provided separately during decryption. In some other embodiments, the AAD is generated at decryption time based at least in part on the request and or other metadata such that decryption will only succeed when the metadata passes. In some embodiments, policy may constrain whether a cryptographic operation can be performed with respect to particular AAD. Processing of Encrypt(KeyID, Data, [AAD]) requests may require, by programming logic and/or policy enforced by the cryptography service, both that the AAD contain particular values and that the AAD be authentic (e.g., not modified since original transmission). Similarly, a Decrypt(KeyID, Ciphertext, [AAD]) request may be used to cause the cryptography service to decrypt the specified ciphertext using a key identified by the KeyID. The AAD in the Decrypt(KeyID, Ciphertext, [AAD]) request may be used such as described above. For instance, processing of the Decrypt(KeyID, Ciphertext, [AAD]) may require, by programming logic and/or policy enforced by the cryptography service, both that the AAD contain particular values and that the AAD be authentic (e.g., not modified since original transmission).

[0088] The Shred(KeyID), in an embodiment, may be used to cause the cryptography service to electronically shred a key or family of keys identified by the specified KeyID. Electronic shredding may include making the key no longer accessible. For example, use of the Shred(KeyID) request may cause the cryptography system to command one or more hardware devices to perform a SecureErase operation on one or more keys identified by the specified KeyID. Generally, the key(s) identified by the KeyID may be electronically shredded in any suitable manner, such as by overwriting data encoding the key with other data (e.g., a series of zeroes or ones or a random string). If the key(s) are stored encrypted under a key, the key used to encrypt the keys may be electronically shredded, thereby causing a loss of access to the key(s). In some embodiments, the shred operation may cause decrypt operations indicating the shredded KeyID to fail at some determined point in the future. Other manners of securely and permanently destroying any possible access to the key(s) may be used.

[0089] The ReKey(Ciphertext, OldKeyID, NewKeyID) request, in an embodiment, may be used to cause the cryptography service to encrypt ciphertext under a different key. When the cryptography service receives a ReKey(Ciphertext, OldKeyID, NewKeyID) request, it may use a key identified by the OldKeyID to decrypt the specified ciphertext and then use a key identified by the NewKeyID to encrypt the decrypted ciphertext. If a key identified by the NewKeyID does not yet exist, the cryptography service may generate a key to use and associate the generated key with the specified NewKeyID, such as described in connection

the Create(KeyID) request described above. In some embodiments, the ReKey operation may be operable to cause data to be transferrable between isolated instances of a cryptography service. In some embodiments, a policy might permit a rekey operation to be performed on a ciphertext but might not permit the same requestor to directly decrypt the ciphertext. In some embodiments, ReKey might support rekeying a ciphertext from a key identified by a first KeyID within a first account to a key identified by a KeyID within a second account.

[0090] Similarly, the frontend system may implement a policy configuration API which, in an embodiment, enables users to submit requests for configuring policies for the performance of cryptographic operations and for other policy-related operations. Policies may be associated with keys, groups of keys, accounts, users and other logical entities in various embodiments. Example policies, which may be configured via the policy configuration API, are provided below. In an embodiment, the cryptography service policy configuration API includes the following requests:

SetKeyPolicy(KeyID, Policy)

Suspend(KeyID, Public Key)

Reinstate(KeyID, Private Key)

[0091] In an embodiment, the SetKeyPolicy(KeyID, Policy) request may be used to cause the cryptography service to store a policy on the key (or family of keys) identified by the KeyID. A policy may be information that is determinative of whether a requested cryptographic operation can be performed in a particular context. The policy may be encoded in a declarative access control policy language, such as eXtensinble Access Control Markup Language (XACML), Enterprise Privacy Authorization Language (EPAL), Amazon Web Services Access Policy Language, Microsoft SecPol or any suitable way of encoding one or more conditions that must be satisfied for a cryptographic operation to be performed. Policies may define what operations can be performed, when the operations can be performed, which entities can make authorized requests for operations to be performed, which information is required for a particular request to be authorized, and the like. In addition, policies may be defined and/or enforced using access control lists, privileges associated with users, and/or operation bitmasks in addition to or instead of the examples given above. Example policies appear below.

[0092] In some embodiments the cryptographic service may support a suspend operation, e.g., using a Suspend(KeyID, Public Key) API call. A suspend operation enables the

customer of the cryptographic service to deny the operator of the cryptographic service use of or access to a key. This can be useful to customers concerned about covert lawful orders or other circumstances in which the operator of the cryptographic service might be compelled to perform some operation using a key. It may also be useful to customers that wish to lock particular data and render it inaccessible online. In some embodiments, a suspend operation might include receiving a public key from a customer and encrypting the key specified by a given KeyID with the received public key and shredding the key specified by the KeyID, such that the provider is not able to access the suspended key unless the private key associated with the public key is provided, e.g., using a Reinstate(KeyID, Private Key) API Call that both specifies the KeyID and includes the private key. In some other embodiments, a suspend operation might involve encrypting a key associated with a specified KeyID using another key managed by the cryptographic service, including without limitation one created for the purpose of the instant suspend operation. The ciphertext produced by this operation can be provided to the customer and not retained within the cryptographic service. The original key identified by the KeyID can then be shredded. The cryptographic service may be operable to receive the provided cipertext and re-import the suspended key. In some embodiments the ciphertext may be generated in a manner that will prevent the cryptographic service from returning a decrypted version to the customer.

[0093] As illustrated in FIG. 16, the cryptography service 1600 includes a backend system that itself comprises various components in some embodiments. For example, the backend system in this example includes a request processing system which may be a subsystem of the cryptography service 1600 that is configured to perform operations in accordance with requests received through either the request API or the policy configuration API. For example, the request processing component may receive requests received via the request API and the policy configuration API determines whether such requests are authentic and are therefore fulfillable and may fulfill the requests. Fulfilling the request may include, for example, performing and/or having performed cryptographic operations. The request processing unit may be configured to interact with an authentication interface which enables the request processing unit to determine whether requests are authentic. The authentication interface may be configured to interact with an authentication system such as described above. For example, when a request is received by the request processing unit, the request processing unit may utilize the authentication interface to interact with an authentication service which may, if applicable, provide authentication proof that may be used in order to cause a performance of cryptographic operations.

[0094] The backend system of the cryptography service 1600 also, in this illustrative example, includes a plurality of security modules (cryptography modules) and a policy enforcement module. One or more of the security modules may be hardware security modules although, in various embodiments, a security module may be any suitable computer device configured according to have capabilities described herein. Each security module in an embodiment stores a plurality of keys associated with KeyIDs. Each security module may be configured to securely store the keys so as to not be accessible by other components of the cryptography service 1600 and/or other components of other systems. In an embodiment, some or all of the security modules are compliant with at least one security standard. For example, in some embodiments, the security modules are each validated as compliant with a Federal Information Processing Standard (FIPS) outlined in FIPS Publication 140-1 and/or 140-2, such as one or more security levels outlined in FIPS Publication 140-2. In addition, in some embodiments, each security module is certified under the Cryptographic Module Validation Program (CMVP). A security module may be implemented as a hardware security module (HSM) or another security module having some or all capabilities of an HSM. In some embodiments, a validated module is used to bootstrap operations. In some embodiments, customers can configure some keys that are stored in and operated on only by validated modules and other keys that are operated on by software. In some embodiments, the performance or cost associated with these various options may differ.

[0095] The security modules may be configured to perform cryptographic operations in accordance with instructions provided by the request processing unit. For example, the request processing unit may provide ciphertext and a KeyID to an appropriate security module with instructions to the security module to use a key associated with the KeyID to decrypt the ciphertext and provide in response the plaintext. In an embodiment, the backend system of the cryptography service 1600 securely stores a plurality of keys forming a key space. Each of the security modules may store all keys in the key space; however, variations are considered as being within the scope of the present disclosure. For example, each of the security modules may store a subspace of the key space. Subspaces of the key space stored by security modules may overlap so that the keys are redundantly stored throughout the security modules. In some embodiments, certain keys may be stored only in specified geographic regions. In some embodiments, certain keys may be accessible only to operators having a particular certification or clearance level. In some embodiments certain keys may be stored in and used only with a module operated by a particular third party provider under contract with the provider of data storage services. In some embodiments, constructive control of security modules may require that lawful orders seeking to compel use of keys

other than as authorized by the customer to involve either additional entities being compelled or additional jurisdictions compelling action. In some embodiments, customers may be offered independent options for the jurisdiction in which their ciphertexts are stored and their keys are stored. In some embodiments, security modules storing keys may be configured to provide audit information to the owner of the keys, and the security modules may be configured such that the generation and providing of audit information not suppressible by the customer. In some embodiments, the security modules may be configured to independently validate a signature generated by the customer such that the provider (e.g., hosting the security modules) is not able to perform operations under keys stored by the security modules. In addition, some security models may store all of the key space and some security modules may store subspaces of the key space. Other variations are also considered as being the scope of the present disclosure. In instances where different security modules store different subspaces of the key space, the request processing unit may be configured such as with a relational table or other mechanism to determine which security module to instruct to perform cryptographic operations in accordance with various requests.

[0096] In an embodiment, the policy enforcement module is configured to obtain information from a request processing unit and determine, based at least in part on that information, whether the request received through the API may be performed. For example, when a request to perform cryptographic operation is received through the request API, the request processing unit may interact with the policy enforcement module to determine whether fulfillment of the request is authorized according to any applicable policy such as policy applicable to a specified KeyID in the request and/or other policies such as policy associated with the requestor. If the policy enforcement module allows fulfillment of the request, the request processing unit may, accordingly, instruct an appropriate security module to perform cryptographic operations in accordance with fulfilling the request.

[0097] As with all figures described herein, numerous variations are considered as being within the scope of the present disclosure. For example, FIG. 16 shows the policy enforcement module separate from security modules. However, each security module may include a policy enforcement module in addition to or instead of the policy enforcement module illustrated as separate. Thus, each security module may be independently configured to enforce policy. In addition, as another example, each security module may include a policy enforcement module which enforces policies different from policies enforced by a separate policy enforcement module. Numerous other variations are considered as being within the scope of the present disclosure.

[0098] As discussed above, various policies may be configured by users in or association with KeyID such that when requests specify cryptographic operations being performed in connection with keys corresponding to KeyIDs, policy may be enforced. FIG. 17 provides an illustrative example of the process 1700 for updating policy in accordance with various embodiments. Process 1700 may be performed by any suitable system, such as by a cryptography service system, such as described above in connection with FIG. 16. In an embodiment, the process 1300 includes receiving 1302 a request to update policy for a KeyID. The request may be received 1302 in any suitable manner. For example, referring to FIG. 16 as an example, a request may be received through a policy configuration API of the frontend system of the cryptography service 1600 described above. The request may be received in any suitable manner.

[0099] The process 1700 in an embodiment includes submitting 1704 an authentication request and receiving 1706 an authentication response. Submitting 1704 the authentication request and receiving 1706 the authentication response may be performed in any suitable manner such as described above. Also as described above, the received authentication response may be used to determine 1708 whether the request to update policy for the KeyID is authentic. If it is determined 1708 that the received request to update policy for the KeyID is not authentic, the request may be denied 1710. Denying 1710 the request may be performed in any suitable manner as described above. If, however, it is determined 1708 that the received request to update policy for the KeyID is authentic, the process 1700 may include accessing 1712 policy information applicable to the requestor. The policy information may be information from which any policy applicable to the requestor may be enforced. For example, within an organization that utilizes a cryptography service performed by process 1700, only certain users of the organization may be allowed to update policy for KeyIDs. Policy information may indicate which users are able to cause the cryptography service to update policy for the KeyID and/or even whether the policy is updatable according to an existing policy. For example, a cryptography service may, in some embodiments, receive request to enforce a new policy. The cryptography service may check whether any existing policy allows the new policy to be put into place. If the cryptography service determines that the existing policy does not allow enforcement of the new policy, the request may be denied. Generally, the policy information may be any information usable for the enforcement of policy applicable to the requestor.

[0100] As illustrated in FIG. 17, the process 1700 includes using the access policy information to determine 1704 whether policy allows the requested update to be performed.

If it is determined 1714 that the policy does not allow the requested update to be performed, the process 1700 may include denying 1710 the request such as described above. If, however, it is determined 1714 the policy allows the requested update to be performed, the process 1700 may include updating 1716 policy for the KeyID. Updating policy for the KeyID may include updating policy information and having the updated policy being stored in accordance with or in association with the KeyID. The updated policy information may be stored, for example, by a policy enforcement module of the cryptography service such as described above, in connection with FIG. 16.

[0101] Policy may also be enforced by other components of an electronic environment that operate in connection with a cryptography service. For instance, referring FIG. 2, discussed above, a cryptography service may provide an electronic representation of a policy to the data service frontend for the data service frontend to enforce. Such may be useful in circumstances where the data service is better suited to enforce the policy. For example, whether an action is allowed by policy may be based at least in part on information accessible to the data service frontend and not to the cryptography service. As one example, a policy may depend on data stored by the data service backend storage system on behalf of a customer associated with the policy.

[0102] As discussed above, a cryptography service may include various systems that allow for the enforcement of policy in accordance with policy on keys having KeyIDs. FIG. 18, accordingly, shows an illustrated example of a process 1800 which may be used to enforce policy. The process 1800 may be performed by any suitable system such as by a cryptography service system such as described above in connection with FIG. 16. In an embodiment, the process 1800 includes receiving 1802 a request to perform one or more cryptographic operations using a key having a KeyID. While FIG. 18 illustrates the process 1800 as being performed in connection with a request to perform one or more cryptographic operations, it should be noted that the process 1800 may be adapted for use with any request to perform an operation which is not necessarily cryptographic. Example operations are described above.

[0103] A determination may be made 1804 whether the received request is authentic. Determining whether the received request is authentic may be performed in any suitable manner such as described above. For instance, determining 1804 whether the request is authentic may include submitting an authentication request and receiving an authentication response such as described above. If it is determined 1804 that the request is not authentic, the process 1800 may include denying 1806 the request. Denying 1806 the request may be

performed in any suitable manner such as described above. If, however, it is determined 1804 that the request is authentic, the process 1800 may include accessing 1808 policy information for the KeyID and/or the requestor. Accessing policy information for the KeyID and/or request may be performed in any suitable manner. For example, accessing policy information for the KeyID and/or requestor may be performed by accessing the storage policy information from one or more storage systems that store such policy information. The access policy information may be used to determine 1810 whether policy allows the one or more operations to be performed.

[0104] If it is determined 1810 that policy does not allow the one or more operations to be performed, the process 1800 may include denying 1806 the request. If, however, it is determined that policy does allow the one or more operations to be performed, the process 1800 may include performing 1812 the requested one or more cryptographic operations. One or more results of performance of the one or more cryptographic operations may be provided 1814 such as provided to the requestor that submitted the received 1802 request to perform the one or more cryptographic operations. In some embodiments information derived at least in part from allowed requests and or denied requests may be provided through an audit subsystem.

[0105] As discussed, embodiments of the present disclosure allow for flexible policy configuration and enforcement. In some embodiments, policies may state which services can perform which operations in which contexts. For example, a policy on a key may allow a data storage service to cause the cryptography service to perform encryption operations but not decryption operations. A policy on a key may also include one or more conditions on the ciphertext and/or decrypted plaintext. For example, a policy may require that the ciphertext and/or plaintext produce a certain hash value (which may be a keyed hash value) before results of an operation be provided in response to a request. Policies may specify one or more restrictions and/or permissions that are based at least in part on time, Internet Protocol (IP) from which requests originate, types of content to be encrypted/decrypted, AAD, and/or other information.

[0106] Numerous variations are considered as being within the scope of the present disclosure. For example, various embodiments discussed above discuss interaction with a separate authentication service. However, components of the environments discussed above may have their own authorization components and determining whether requests are authentic may or may not involve communication with another entity. Further, each of the environments discussed above are illustrated in connection with particular operations and

capabilities enabled by the environments. The techniques discussed above in connection with different environments may be combined and, generally, environments in accordance with the present disclosure may allow for flexible use of the various techniques. As just one example, a cryptography service may be used to, upon request, encrypt both keys and other content, such as non-key data objects. As another example, a cryptography service may be configured to receive and respond to requests from both users (e.g., customers of a computing resource provider) and other services (e.g., data storage services). In some embodiments, a cryptography service and/or associated authentication service may be configured for use with a mobile device to perform encryption of stored data. In some embodiments at least one unlock pin may be validated by a cryptography service. In still other embodiments a cryptographic service, as part of an operation, may receive information generated by a hardware attestation. In some embodiments a cryptography service may be operable to provide digital rights management services with respect to content.

[0107] Embodiments of the disclosure can be described in view of the following clauses:

1. A computer-implemented method for providing data storage services, comprising:

under the control of one or more computer systems of a computing resource service provider, the one or more computer systems configured with executable instructions,

receiving, from a customer of the computing resource service provider, a request to utilize a data storage service of the computing resource service provider;

as a result of having received the request to utilize the data storage service, causing a cryptography service of the computing resource service provider to provide information encrypted by the cryptography service using a key that is inaccessible to the data storage service, the information usable to obtain the data object in unencrypted form and the key from a plurality of keys managed by the cryptography service on behalf of a plurality of customers of the computing resource service provider; and

using the data storage service to store the encrypted information.

2. The computer-implemented method of clause 1, wherein:

causing the cryptography service to provide the encrypted information includes providing proof information to the cryptography service that enables the cryptography service to verify authenticity of the received request; and

the proof information is necessary for the cryptography service to provide the encrypted information.

3. The computer-implemented method of clause 1 or 2, wherein:

the request to utilize the data storage service includes a request to store, in encrypted form, a data object using the data storage service;

the information includes a second key used to encrypt the data object; and

using the data storage service to store the encrypted information includes using the data storage service to store, in encrypted form, the data object.

4. The computer-implemented method of any one of the preceding clauses, wherein the method further comprises:

receiving a request to retrieve the data object from the data storage service;

obtaining the encrypted information from the data storage service;

causing the cryptography service to decrypt the encrypted information; and

using the information to provide the data object in response to the received request to retrieve the data object.

5. The computer-implemented method of clause 4, wherein using the information to provide the data object in response to the received request to retrieve the data object includes using the key to decrypt the encrypted data object.

6. The computer-implemented method of any one of the preceding clauses, wherein the method further comprises:

providing to the cryptography service the encrypted information for decryption; and

causing the cryptography service to check whether a policy on the key allows the key to be used to decrypt the encrypted information and, when allowed by the policy, decrypt the encrypted information.

7. The computer-implemented method of any one of the preceding clauses, wherein:

the method further comprises receiving, from the customer, information representing a policy on the key;

causing the cryptography service to operate in accordance with the represented policy.

8. The computer-implemented method of any one of the preceding clauses, wherein:

the method further comprises:

receiving, from the cryptography service, a policy on the key;

checking whether the received policy allows fulfilling the request; and

causing the cryptography service to provide the encrypted information is dependent on the policy allows fulfilling the request.

causing the cryptography service to operate in accordance with the represented policy.

9. A computer-implemented method for providing cryptographic services, comprising:

under the control of one or more computer systems configured with executable instructions,

as a result of a service receiving a first request to utilize the service, receiving a second request to perform one or more cryptographic operations that are necessary for fulfilling the first request;

performing the requested one or more cryptographic operations; and

providing one or more results of performing the requested one or more cryptographic operations to the service, the one or more results necessary for at least one manner of fulfilling the received request to utilize the service.

10. The computer-implemented method of clause 9, wherein the one or more cryptographic operations include decryption of a key needed to fulfill the received request to utilize the data service.

11. The computer-implemented method of clause 9 or 10, wherein:

the method further comprises further comprising receiving information verifying that the received request to utilize the data service is authentic; and

performing the one or more cryptographic operations requires receipt of the information verifying that the received request to utilize the service is authentic.

12. The computer-implemented method of any one of clauses 9 to 11, wherein the received request to utilize the service is a request to obtain data stored by the service.

13. The computer-implemented method of any one of clauses 9 to 12, wherein the received request to utilize the service is a request to store data using the service.

14. The computer-implemented method of any one of clauses 9 to 13, further comprising selecting, from a plurality of keys that comprises at least two keys each managed

by the cryptography service on behalf of a different entity, at least one key for performing the one or more cryptographic operations.

15. The computer-implemented method of any one of clauses 9 to 14, wherein:
the one or more cryptographic operations require use of a key;
the method further comprises receiving, for the key, a policy for the key; and
performing the one or more cryptographic operations requires performance of the one or more cryptographic operations to be in compliance with the received policy.

16. The computer-implemented method of any one of clauses 9 to 15, further comprising:
receiving a policy for a key usable to perform the one or more cryptographic operations;
checking whether an existing policy allows implementation of the policy; and
rejecting the received policy as a result of the existing policy disallowing implementation of the policy.

17. A computer system, comprising:
one or more processors; and
memory storing instructions executable by the one or more processors to cause the computer system to implement at least:
a cryptography service configured to at least:
store a plurality of keys such that the plurality of keys are inaccessible to a service different from the cryptography service; and
upon detecting a request pending at the service, selecting a key from the plurality of keys and using the selected key to perform one or more cryptographic operations needed to fulfill the pending request.

18. The computer system of clause 17, wherein detecting the request pending at the service includes receiving, from the service, notification and proof of the request pending at the service.

19. The computer system of clause 17 or 18, wherein:
the computer system is operated by a computing resource service provider;

the request is generated by a customer of a plurality of customers of the computing resource provider; and

each key of the plurality of keys has a corresponding customer of the computing resource provider.

20. The computer system of any one of clauses 17 to 19, wherein the cryptography service is further configured to verify that the pending request is authorized.

21. The computer system of any one of clauses 17 to 20, further comprising a metering service configured to receive output of the cryptography service and generate accounting records based at least in part on the received output.

22. The computer system of any one of clauses 17 to 21, wherein the cryptography service is further configured to:

store policy information that defines one or more policies for each of at least a subset of the plurality of keys; and

enforce the defined one or more policies.

23. The computer system of clause 22, wherein:

the computer system is operated by a computing resource service provider;

each key of the plurality of keys corresponds to a customer of the computing resource provider; and

the cryptography service is configured to:

receive, from customers of the computing resource provider, policy updates; and

update the stored policy information according to the received policy updates.

24. A computer-readable storage medium having stored thereon instructions that, when executed by one or more processors of a computer system, cause the computer system to at least:

receive a request to utilize a service;

as a result of having received the request to utilize the service:

provide to a cryptography service proof, based at least in part on the request, that the request was received, thereby causing the cryptography service to use a key to perform one or more cryptographic operations on information that, after the one or more cryptographic operations have been performed, is usable to fulfill the request to utilize the service; and

fulfill the request to utilize the service using the information on which the one or more cryptographic operations were performed.

25. The computer-readable storage medium of clause 24, wherein the proof comprises authentication information that enables the cryptography service to verify that the request to utilize the service is authentic.

26. The computer-readable storage medium of clause 24 or 25, wherein:

the request to utilize the service includes a request to perform an operation in connection with a data object; and

the information is a key used to encrypt the data object.

27. The computer-readable storage medium of any one of clauses 24 to 26, wherein the instructions, when executed by the one or more processors, cause the cryptography service to enforce one or more policies on the key, the one or more policies being determinative of whether the one or more cryptographic operations are performable using the key.

28. The computer-readable storage medium of any one of clauses 24 to 27, wherein the request to utilize the service is a request to perform a storage operation in connection with a data object.

29. The computer-readable storage medium of any one of clauses 24 to 28, wherein causing the cryptography service to use the key to perform one or more cryptographic operations includes providing an identifier of the key to enable the cryptography service to select the key from a plurality of keys stored by the cryptography service.

[0108] FIG. 19 illustrates aspects of an example environment 1900 for implementing aspects in accordance with various embodiments. As will be appreciated, although a Web-based environment is used for purposes of explanation, different environments may be used, as appropriate, to implement various embodiments. The environment includes an electronic client device 1902, which can include any appropriate device operable to send and receive requests, messages or information over an appropriate network 1904 and convey information back to a user of the device. Examples of such client devices include personal computers, cell phones, handheld messaging devices, laptop computers, set-top boxes, personal data assistants, electronic book readers and the like. The network can include any appropriate

network, including an intranet, the Internet, a cellular network, a local area network or any other such network or combination thereof. Components used for such a system can depend at least in part upon the type of network and/or environment selected. Protocols and components for communicating via such a network are well known and will not be discussed herein in detail. Communication over the network can be enabled by wired or wireless connections and combinations thereof. In this example, the network includes the Internet, as the environment includes a Web server 1906 for receiving requests and serving content in response thereto, although for other networks an alternative device serving a similar purpose could be used as would be apparent to one of ordinary skill in the art.

[0109] The illustrative environment includes at least one application server 1908 and a data store 1910. It should be understood that there can be several application servers, layers, or other elements, processes or components, which may be chained or otherwise configured, which can interact to perform tasks such as obtaining data from an appropriate data store. As used herein the term "data store" refers to any device or combination of devices capable of storing, accessing and retrieving data, which may include any combination and number of data servers, databases, data storage devices and data storage media, in any standard, distributed or clustered environment. The application server can include any appropriate hardware and software for integrating with the data store as needed to execute aspects of one or more applications for the client device, handling a majority of the data access and business logic for an application. The application server provides access control services in cooperation with the data store, and is able to generate content such as text, graphics, audio and/or video to be transferred to the user, which may be served to the user by the Web server in the form of HyperText Markup Language ("HTML"), Extensible Markup Language ("XML") or another appropriate structured language in this example. The handling of all requests and responses, as well as the delivery of content between the client device 1902 and the application server 1908, can be handled by the Web server. It should be understood that the Web and application servers are not required and are merely example components, as structured code discussed herein can be executed on any appropriate device or host machine as discussed elsewhere herein.

[0110] The data store 1910 can include several separate data tables, databases or other data storage mechanisms and media for storing data relating to a particular aspect. For example, the data store illustrated includes mechanisms for storing production data 1912 and user information 1916, which can be used to serve content for the production side. The data store also is shown to include a mechanism for storing log data 1914, which can be used for

reporting, analysis or other such purposes. It should be understood that there can be many other aspects that may need to be stored in the data store, such as for page image information and to access right information, which can be stored in any of the above listed mechanisms as appropriate or in additional mechanisms in the data store 1910. The data store 1910 is operable, through logic associated therewith, to receive instructions from the application server 1908 and obtain, update or otherwise process data in response thereto. In one example, a user might submit a search request for a certain type of item. In this case, the data store might access the user information to verify the identity of the user, and can access the catalog detail information to obtain information about items of that type. The information then can be returned to the user, such as in a results listing on a Web page that the user is able to view via a browser on the user device 1902. Information for a particular item of interest can be viewed in a dedicated page or window of the browser.

[0111] Each server typically will include an operating system that provides executable program instructions for the general administration and operation of that server, and typically will include a computer-readable storage medium (e.g., a hard disk, random access memory, read only memory, etc.) storing instructions that, when executed by a processor of the server, allow the server to perform its intended functions. Suitable implementations for the operating system and general functionality of the servers are known or commercially available, and are readily implemented by persons having ordinary skill in the art, particularly in light of the disclosure herein. In some embodiments, an operating system may be configured in accordance with or validated under one or more validation regimes such as Evaluation Assurance Level (EAL) level 4.

[0112] The environment in one embodiment is a distributed computing environment utilizing several computer systems and components that are interconnected via communication links, using one or more computer networks or direct connections. However, it will be appreciated by those of ordinary skill in the art that such a system could operate equally well in a system having fewer or a greater number of components than are illustrated in FIG. 19. Thus, the depiction of the system 1900 in FIG. 19 should be taken as being illustrative in nature, and not limiting to the scope of the disclosure.

[0113] The various embodiments further can be implemented in a wide variety of operating environments, which in some cases can include one or more user computers, computing devices or processing devices which can be used to operate any of a number of applications. User or client devices can include any of a number of general purpose personal computers, such as desktop or laptop computers running a standard operating system, as well as cellular,

wireless and handheld devices running mobile software and capable of supporting a number of networking and messaging protocols. Such a system also can include a number of workstations running any of a variety of commercially-available operating systems and other known applications for purposes such as development and database management. These devices also can include other electronic devices, such as dummy terminals, thin-clients, gaming systems and other devices capable of communicating via a network.

[0114] Most embodiments utilize at least one network that would be familiar to those skilled in the art for supporting communications using any of a variety of commercially-available models and protocols, such as Transmission Control Protocol/Internet Protocol ("TCP/IP"), Open System Interconnection ("OSI"), File Transfer Protocol ("FTP"), Universal Plug and Play ("UpnP"), Network File System ("NFS"), Common Internet File System ("CIFS") and AppleTalk. The network can be, for example, a local area network, a wide-area network, a virtual private network, the Internet, an intranet, an extranet, a public switched telephone network, an infrared network, a wireless network and any combination thereof.

[0115] In embodiments utilizing a Web server, the Web server can run any of a variety of server or mid-tier applications, including Hypertext Transfer Protocol ("HTTP") servers, FTP servers, Common Gateway Interface ("CGI") servers, data servers, Java servers and business application servers. The server(s) also may be capable of executing programs or scripts in response requests from user devices, such as by executing one or more Web applications that may be implemented as one or more scripts or programs written in any programming language, such as Java®, C, C# or C++, or any scripting language, such as Perl, Python or TCL, as well as combinations thereof. The server(s) may also include database servers, including without limitation those commercially available from Oracle®, Microsoft®, Sybase® and IBM®.

[0116] The environment can include a variety of data stores and other memory and storage media as discussed above. These can reside in a variety of locations, such as on a storage medium local to (and/or resident in) one or more of the computers or remote from any or all of the computers across the network. In a particular set of embodiments, the information may reside in a storage-area network ("SAN") familiar to those skilled in the art. Similarly, any necessary files for performing the functions attributed to the computers, servers or other network devices may be stored locally and/or remotely, as appropriate. Where a system includes computerized devices, each such device can include hardware elements that may be electrically coupled via a bus, the elements including, for example, at least one central processing unit ("CPU"), at least one input device (*e.g.*, a mouse, keyboard, controller, touch

screen or keypad), and at least one output device (*e.g.*, a display device, printer or speaker). Such a system may also include one or more storage devices, such as disk drives, optical storage devices, and solid-state storage devices such as random access memory ("RAM") or read-only memory ("ROM"), as well as removable media devices, memory cards, flash cards, etc. Various embodiments of the present disclosure may also be implemented using custom hardware including, but not limited, custom cryptographic processors, smart cards and/or hardware security modules.

[0117] Such devices also can include a computer-readable storage media reader, a communications device (*e.g.*, a modem, a network card (wireless or wired), an infrared communication device, etc.) and working memory as described above. The computer-readable storage media reader can be connected with, or configured to receive, a computer-readable storage medium, representing remote, local, fixed and/or removable storage devices as well as storage media for temporarily and/or more permanently containing, storing, transmitting and retrieving computer-readable information. The system and various devices also typically will include a number of software applications, modules, services or other elements located within at least one working memory device, including an operating system and application programs, such as a client application or Web browser. It should be appreciated that alternate embodiments may have numerous variations from that described above. For example, customized hardware might also be used and/or particular elements might be implemented in hardware, software (including portable software, such as applets) or both. Further, connection to other computing devices such as network input/output devices may be employed.

[0118] Storage media and computer readable media for containing code, or portions of code, can include any appropriate media known or used in the art, including storage media and communication media, such as but not limited to volatile and non-volatile, removable and non-removable media implemented in any method or technology for storage and/or transmission of information such as computer readable instructions, data structures, program modules or other data, including RAM, ROM, Electrically Erasable Programmable Read-Only Memory ("EEPROM"), flash memory or other memory technology, Compact Disc Read-Only Memory ("CD-ROM"), digital versatile disk (DVD) or other optical storage, magnetic cassettes, magnetic tape, magnetic disk storage or other magnetic storage devices or any other medium which can be used to store the desired information and which can be accessed by the a system device. Based on the disclosure and teachings provided herein, a

person of ordinary skill in the art will appreciate other ways and/or methods to implement the various embodiments.

[0119] The specification and drawings are, accordingly, to be regarded in an illustrative rather than a restrictive sense. It will, however, be evident that various modifications and changes may be made thereunto without departing from the broader spirit and scope of the invention as set forth in the claims.

[0120] Other variations are within the spirit of the present disclosure. Thus, while the disclosed techniques are susceptible to various modifications and alternative constructions, certain illustrated embodiments thereof are shown in the drawings and have been described above in detail. It should be understood, however, that there is no intention to limit the invention to the specific form or forms disclosed, but on the contrary, the intention is to cover all modifications, alternative constructions and equivalents falling within the spirit and scope of the invention, as defined in the appended claims.

[0121] The use of the terms "a" and "an" and "the" and similar referents in the context of describing the disclosed embodiments (especially in the context of the following claims) are to be construed to cover both the singular and the plural, unless otherwise indicated herein or clearly contradicted by context. The terms "comprising," "having," "including," and "containing" are to be construed as open-ended terms (i.e., meaning "including, but not limited to,") unless otherwise noted. The term "connected" is to be construed as partly or wholly contained within, attached to, or joined together, even if there is something intervening. Recitation of ranges of values herein are merely intended to serve as a shorthand method of referring individually to each separate value falling within the range, unless otherwise indicated herein, and each separate value is incorporated into the specification as if it were individually recited herein. All methods described herein can be performed in any suitable order unless otherwise indicated herein or otherwise clearly contradicted by context. The use of any and all examples, or exemplary language (*e.g.*, "such as") provided herein, is intended merely to better illuminate embodiments of the invention and does not pose a limitation on the scope of the invention unless otherwise claimed. No language in the specification should be construed as indicating any non-claimed element as essential to the practice of the invention.

[0122] Preferred embodiments of this disclosure are described herein, including the best mode known to the inventors for carrying out the invention. Variations of those preferred embodiments may become apparent to those of ordinary skill in the art upon reading the

foregoing description. The inventors expect skilled artisans to employ such variations as appropriate, and the inventors intend for the invention to be practiced otherwise than as specifically described herein. Accordingly, this invention includes all modifications and equivalents of the subject matter recited in the claims appended hereto as permitted by applicable law. Moreover, any combination of the above-described elements in all possible variations thereof is encompassed by the invention unless otherwise indicated herein or otherwise clearly contradicted by context.

[0123] All references, including publications, patent applications and patents, cited herein are hereby incorporated by reference to the same extent as if each reference were individually and specifically indicated to be incorporated by reference and were set forth in its entirety herein.