

DATA SECURITY SYSTEM AND METHOD FOR OPERATION THEREOF

TECHNICAL FIELD

The present invention relates to a data security system.

More particularly, the present invention relates to a data security system relating to
5 attribute based encrypted ciphertexts.

BACKGROUND ART

Traditionally, access controls to data stored by a hosting service are enforced by
employing trusted servers to store the data and mediate access control to that data.
However, hosting services are increasingly storing the data across multiple servers that
10 are also shared with other data owners. An example of this is cloud computing, which
provides data owners with the capability to store their data on third party servers so that
the data can be accessed by themselves or by selected users. Cloud computing has
great potential in providing various services at significantly reduced cost due to the
aggregated management of elastic resources.

15 Unfortunately there are a number of security risks that may arise in both the software
systems and in the hardware platforms used in cloud computing. When a data owner
elects to store data on a public cloud, it loses its ability to have physical access to the
servers hosting its data. This leads to the data being at risk and being made available to
undesired third parties. To mitigate users' privacy concerns about their data, a common
20 solution is to store the data in encrypted form so that it will remain private even if data
servers become compromised. Encryption is the process of encoding the data in such a
way that only authorized users can decrypt and read the data. As data is more often
being shared amongst many groups of users, the encrypted data should be amenable to
sharing and access control via group policies.

25 Recently, ciphertext-policy attribute-based encryption (CP-ABE) was proposed as a
solution to enforce access control of encrypted data in untrusted servers, such as public
cloud servers, where users with appropriate credentials can access encrypted data based
on security policies set out by the data owners. Unlike the traditional public key
encryption schemes where encryption is performed under a public key and the ciphertext

is decrypted using a single private key, CP-ABE is a one-to-many public key encryption technique, allowing data to be encrypted under a certain access policy based on user attributes instead of their specific identities.

However, standard CP-ABE schemes have two drawbacks which tend to hinder their
5 widespread adoption. The first drawback is that decryption is expensive for resource-limited devices due to pairing operations, and the number of pairing operations required to decrypt a ciphertext grows with the complexity of the access policy. The second drawback is that they assume the users credentials are static, i.e. once a user's attributes are issued by a trusted authority the attributes do not change. However, in the real world
10 the attributes often do change, for example due to a user changing job function or terminating employment. In such instances the user's attributes or decryption key must be revoked or updated to prevent subsequent unauthorized disclosure of sensitive information.

The standard CP-ABE schemes allow scalable sharing of encrypted data stored in the
15 cloud. Every user is issued a decryption key associated with a set of attributes and the ciphertext (i.e. encrypted data) is associated with an access policy based on the users' attribute set. Thus a user can only decrypt the ciphertext if their attribute set satisfies the access policy. For example, a data owner using CP-ABE encrypts an electronic medical record and attaches the access policy (e.g. "Cardiologist" AND "General Hospital") to the
20 ciphertext. Then, any users with the attributes "Cardiologist" and "General Hospital" can decrypt the ciphertext to read the electronic medical record. All other users who do not possess those same attributes are prevented from decrypting the ciphertext. However, the standard CP-ABE scheme has difficulties in handling changes to or revocation of the user attributes. It also tends to be inefficient as the size of the ciphertext and time taken
25 to decrypt it increases with the complexity of the CP-ABE formula.

The latter difficulty described above is overcome by providing CP-ABE with outsourced decryption, wherein a public server is allowed to assist with the decryption operations for users. Thus most of the decryption operations are performed by a server in the public cloud resulting in a partially decrypted ciphertext and thereafter only lightweight decryption
30 operations are performed by a user to recover the plaintext data. An example of such outsourced decryption is described in US9049023. However, such outsourcing to the decryption does not alleviate the difficulty of handling changes to the user attributes.

Various attempts have been made to address the problem of handling changes to the user attributes. For example, user revocation may be performed by a server which stores a server-side secret key being provided for every user. However, data security and user privacy can become compromised if the server-side secret keys are leaked to third parties
 5 or if the hosting server management colludes with revoked users.

It is to be understood that, if any prior art publication is referred to herein, such reference does not constitute an admission that the publication forms a part of the common general knowledge in the art in any country.

SUMMARY OF THE INVENTION

- 10 According to one aspect, there is provided a data security system comprising:
- a public computer server being arranged to store encrypted data as ciphertext;
 - a public database provided on the public computer server being arranged to store a public decryption key for partially decrypting the ciphertext to yield a partially decrypted ciphertext, to store an attribute set arranged to regulate use of the public decryption key,
 - 15 and to store a user identifier to be associated with a user;
 - a private computer server;
 - a private database provided on the private computer server being arranged to store the user identifier, and to store a first private decryption key for partially decrypting the partially decrypted ciphertext to yield an intermediate decrypted ciphertext;
 - 20 a second private decryption key arranged to be provided to the user for fully decrypting the intermediate decrypted ciphertext to yield unencrypted data; and
 - wherein the respective decrypting of the ciphertext and the partially decrypted ciphertext is only permitted if the user identifier is stored in the public database and in the private database.
- 25 The data security system may comprise a key generation program being arranged to generate cryptographic keys including the public decryption key, the first private decryption key, and the second private decryption key.
- The key generation program may be arranged to simultaneously amend the public decryption key and user identifier in the public database, and to amend the first private
 30 decryption key and user identifier in the private database.

The key generation program may be arranged to simultaneously delete the public decryption key and user identifier in the public database, and to delete the first private decryption key and user identifier in the private database.

5 The second private decryption key may be unable to decrypt the ciphertext or to decrypt the partially decrypted ciphertext.

The public computer server may be a public cloud server, an unsecured publically accessible personal computer or an unsecured publically accessible mobile device.

According to another aspect, there is provided a method for operating a data security system comprising the steps of:

- 10 providing encrypted data as ciphertext on a public computer server;
- providing a public decryption key to partially decrypt the ciphertext on the public computer server to yield a partially decrypted ciphertext, wherein the public decryption key is associated with an attribute set arranged to regulate use of the public decryption key;
- transmitting the partially decrypted ciphertext to a private computer server having a
- 15 first private decryption key to further partially decrypt the partially decrypted ciphertext to yield an intermediate decrypted ciphertext; and
- transmitting the intermediate decrypted ciphertext to a user having a second private decryption key to fully decrypt the intermediate decrypted ciphertext to yield unencrypted data;
- 20 wherein the public decryption key and the first private decryption key are associated with a user identifier of the user; and
- wherein the steps of partially decrypting the ciphertext and partially decrypting the partially decrypted ciphertext are only performed if the user identifier is associated with both the public decryption key and the first private decryption key.

25 The public computer server may be a public cloud server, an unsecured publically accessible personal computer or an unsecured publically accessible mobile device.

The step of partially decrypting the ciphertext to yield a partially decrypted ciphertext may comprise decrypting the ciphertext to a substantial extent whereby each of the subsequent steps of decrypting the partially decrypted ciphertext and decrypting the

30 intermediate decrypted ciphertext, respectively comprise performing only one exponential operation.

The minimal computations may comprise performing only one exponential operation.

The method may comprise the step of amending the public decryption key and user identifier on the public computer server, and amending the first private decryption key and user identifier on the private computer server.

- 5 The method may comprise the step of deleting the public decryption key and user identifier from the public computer server, and deleting the first private decryption key and user identifier from the private computer server.

BRIEF DESCRIPTION OF THE DRAWINGS

10 The present invention will now be described, by way of example, with reference to the accompanying schematic drawings, in which:

Figure 1 is a diagrammatic representation of a data security system according to an embodiment of the present invention; and

Figure 2 is an operational flow diagram showing steps in decrypting encrypted data stored on a public cloud.

15 **DETAILED DESCRIPTION OF SPECIFIC EMBODIMENTS**

Referring to the drawings, there is shown a data security system 10 that is arranged to enable users (individuals or organizations) to publish their attribute based encrypted (ABE) data to an untrusted server such as a public cloud 12, whereby the decryption of the data is outsourced to a third party. It will be appreciated that although the description below refers to a server in the public cloud 12, the technology can also be applied in other systems and servers where data is encrypted by one party for being decrypted and read by another party. Thus in the context of this disclosure any reference to untrusted servers refers to servers that are not trusted to keep secret information and can be public cloud servers, individual computers or mobile devices. For example, in an electronic medical record system, an individual's electronic medical record can be encrypted and stored on a private smartphone for emergency access by medical staff directly accessing that smartphone.

The system 10 supports scalable access to the encrypted data since encryption and access policies are specified in terms of a user's attributes. The system 10 further

supports outsourced decryption so that resource constrained devices can be used to access the data. Outsourcing the decryption enables a substantial part of the decryption to be conducted on an untrusted server so that decryption can be performed relatively fast, even on smartphones having constrained computing resources. The untrusted
5 server assists in the decryption without getting any information on the underlying plaintext.

The system 10 also allows efficient user management by enabling attribute updating and user revocation, and provides strong security by resisting server-user collusion attacks. The strong security protection ensures that any compromise of the public cloud servers will not reveal the encrypted data to third parties. Thus even should there be collusion
10 amongst public cloud servers and revoked users they will be unable to decrypt the encrypted data or access the information therein. Obsolete keys (i.e. decryption keys of revoked users) are rendered useless as soon as their owners are revoked from the system.

As shown in Figure 1, the system 10 includes various parties that are able to
15 communicate with each other via a communications network, such as a LAN, WAN or the Internet. The system 10 includes the public cloud 12 in which data 14 is hosted, such as on a cloud based hosting platform. Typically the data 14 will be encrypted and published by data owners 16 to be decrypted and read by selective users 18. Although not shown in Figure 1, it will be appreciated that the data owners 16 and the users 18 will interact with
20 the public cloud 12 via their own computer equipment, which could be regular desktop or laptop personal computers and workstations. However, it is envisaged that the users 18 will also make use of less powerful computer equipment, such as personal digital assistants (PDA's) or mobile phones, in particular smartphones. Typically such smartphones will have constrained computing resources in comparison to personal
25 computers, having slower processors and less memory. It will be appreciated that smartphones also have limited battery capacity and that intensive processor use can quickly drain their batteries. It is thus advantageous to limit processor use where possible to extend the battery life.

The public cloud 12 is arranged to assist in performing partial decryption of the data 14 to
30 alleviate the decryption burden on the computer equipment of the users 18. The data owners 16 specify attribute based access policies comprising various attributes by which the data 14 can be decrypted, which access policies are also uploaded to the public cloud 12.

The system 10 further includes a key generation centre 20 and a management server 22. The key generation centre 20 is responsible for managing and storing user identifiers and a set of attributes associated with each of the users 18. The key generation centre 20 is further arranged to generate public parameters and various cryptographic keys for use in the system 10. The management server 22 is responsible for storing user information relevant to the users 18 received from the key generation centre 20 and to update any changes to the user information in real-time. Both the key generation centre 20 and the management server 22 are normally operated by the data owners 16. However, it is envisioned that the key generation centre 20 and/or the management server 22 could be operated by a trusted third party authority. The public cloud 12 also stores the user information relevant to the users 18 received from the key generation centre 20 and updates any changes to the user information in real-time.

In use, when data 14 is to be published to the public cloud 12, the key generation centre 20 runs a setup algorithm to generate the requisite cryptographic keys. The setup algorithm takes as input a security parameter λ and an attribute universe description U . The attribute universe refers to the set of all attributes recognised by the key generation centre 20. The security parameter λ is an integer and, in general, increasing the security parameter increases the level of security and makes it more difficult for a hacker to derive the decryption key. The setup algorithm outputs a public key PK (also known as the public parameters) and a master secret key MSK .

Whenever a new user 18 joins the system 10 or a user attribute of an existing user 18 is updated, the key generation centre 20 runs a key generation algorithm. The key generation algorithm takes as input the public key PK , the master secret key MSK , a user identifier u , and the user's set of attributes S . The key generation algorithm outputs a secret decryption key SDK , a secret management key SMK and a public decryption key PDK . Each of the secret decryption key SDK , the secret management key SMK and the public decryption key PDK are specifically allocated to the user identifier u of a specific user 18. The secret management key SMK is arranged to be used as a first private decryption key, whereas the secret decryption key SDK is arranged to be used as a second private decryption key. The secret decryption key SDK is sent to the specific user 18 over secure protected channels (indicated by arrow a3 in Figure 1). Both the user identifier u and the secret management key SMK are sent to the management server 22 over secure protected channels for storage in a management server list $MS-L$ (indicated by arrow a2 in Figure 1). The user identifier u , users set of attributes S and the public

decryption key PDK are sent to the public cloud 12 over an authenticated channel for storage in a public cloud list $PC-L$ (indicated by arrow a1 in Figure 1). The public key PK is made available to all the parties in the system 10, including the public cloud 12, the data owners 16, the users 18 and the management server 22.

5 The management server 22 maintains the management server list $MS-L$ in which every active user 18 has an entry that contains the particular user identifier u and the secret management key SMK . The management server 22 receives this twofold entry (u, SMK) from the key generation centre 20 whenever a new user 18 joins the system or a set of attributes of an existing user 18 is updated. If the user 18 is a new user, the twofold entry
 10 (u, SMK) is added to management server list $MS-L$. If the user 18 is an existing user whose set of attributes has changed, the new twofold entry (u, SMK) replaces the existing twofold entry for that user 18. If the user 18 is to be removed from the system 10 so that they cannot decrypt and read any of the data 14, i.e. all their access permissions are revoked, the user's corresponding twofold entry (u, SMK) is deleted from the management
 15 server list $MS-L$.

The public cloud 12 maintains the public cloud list $PC-L$ in which every active user 18 has an entry that contains the particular user identifier u , the user's set of attributes S and the public decryption key PDK . The public cloud 12 receives this threefold entry (u, S, PDK) from the key generation centre 20 whenever a new user 18 joins the system or a set of
 20 attributes of an existing user 18 is updated. If the user 18 is a new user, the threefold entry (u, S, PDK) is added to public cloud list $PC-L$. If the user 18 is an existing user whose set of attributes has changed, the new threefold entry (u, S, PDK) replaces the existing threefold entry for that user 18. If the user 18 is to be removed from the system 10 so that they cannot decrypt and read any of the data 14, i.e. all their access
 25 permissions are revoked, the user's corresponding threefold entry (u, S, PDK) is deleted from the public cloud list $PC-L$.

When a data owner 16 intends publishing a document M (data 14) to the public cloud 12, the data owner 16 runs an encryption algorithm to encrypt the document M from plaintext into ciphertext. The encryption algorithm takes as input the public key PK , the data
 30 document M and an access policy T , wherein the access policy T specifies the required attribute set that users must have in order to be permitted to read the document M (indicated by arrow b1 in Figure 1). The encryption algorithm outputs a ciphertext CT .

The data owner 16 then sends both the ciphertext CT and its related access policy T to the public cloud 12 for storage (indicated by arrow b2 in Figure 1).

The public cloud 12 stores the ciphertext CT and its access policy T in a storage system on one or more its servers.

5 The operational steps in decrypting the ciphertext as explained below are also illustrated in the flow diagram shown in Figure 2. When the public cloud 12 receives a request from a user 18 to access the ciphertext CT (indicated by arrow c in Figure 1), the public cloud runs a primary decryption algorithm. The primary decryption algorithm takes as input the public key PK , the ciphertext CT , the access policy T , the user identifier u , the user's set of attributes S and the public decryption key PDK . In a first step, the public cloud 12
10 conducts a cross check to determine if the user identifier u exists in the public cloud list $PC-L$. If the user identifier u does not exist, then the public cloud 12 rejects the request. Otherwise, the public cloud 12 conducts a further test to determine if the user's set of attributes S correlates to and satisfies the access policy T . If not, the public
15 cloud 12 rejects the request of the user 18; if yes, the public cloud 12 partially decrypts the ciphertext CT using the public decryption key PDK to generate a partially decrypted ciphertext $PDCT$. The primary decryption algorithm is designed to perform most of the computations necessary to decrypt the ciphertext CT so as to take advantage of the abundant computational powers available in the public cloud 12. Thereafter the public
20 cloud 12 sends both the user identifier u and the partially decrypted ciphertext $PDCT$ to the management server 22 (indicated by arrow d1 in Figure 1).

As an example, assume that the user's set of attributes $S = (\text{"Cardiologist", "General Hospital", "IC\#456789", "Alice Chen"})$ and further that the access policy T associated with a particular ciphertext CT is $T = (\text{"Cardiologist" AND "General Hospital"})$ OR $(\text{"John Smith" AND "IC\#123456"})$. It can be seen that the user set of attributes S does satisfy the access
25 policy T due to containing both "Cardiologist" and "General Hospital". In comparison, another user 18 having the user's set of attributes $S = (\text{"Cardiologist", "University Hospital"})$ would not satisfy the access policy T .

After the management server 22 receives the partially decrypted ciphertext $PDCT$ from
30 the public cloud 12, the management server 22 runs a secondary decryption algorithm. The secondary decryption algorithm takes as input the public key PK , the partially decrypted ciphertext $PDCT$, the user identifier u , and the user's secret management

key SMK . In a first step, the management server 22 conducts a cross check to determine if the user identifier u exists in the management server list $MS-L$. If the user identifier u does not exist, then the management server 22 rejects the request. Otherwise, the management server 22 further decrypts the partially decrypted ciphertext $PDCT$ using the
 5 secret management key SMK to generate an intermediate decrypted ciphertext $IDCT$, which is sent to the user 18 (indicated by arrow d2 in Figure 1).

Upon the user 18 receiving the intermediate decrypted ciphertext $IDCT$, the user 18 runs a final decryption algorithm. The final decryption algorithm takes as input the public key PK , the intermediate decrypted ciphertext $IDCT$, and the secret decryption key SDK . The final
 10 decryption algorithm outputs the decrypted document M in plaintext so that it can be read by the user 18 (indicated by d3 in Figure 1).

As most of the decryption is performed by the public cloud 12, it will be appreciated that both the secondary and final decryption algorithms perform lightweight operations so that they can be carried out efficiently even on constrained computing devices.

15 It will be appreciated that the following assumptions are made regarding the various parties in the system 10:

- The public cloud 12 is an untrusted server in the sense that it is cannot guarantee keeping information (the data 14) secret. Thus the public cloud 12 stores the encrypted ciphertext CT and its associated access policy T , the user identifiers u and
 20 each user's set of attributes S and the public decryption key PDK , which are all assumed to be public information. Thus there is a possibility that the public cloud 12 (i.e. people controlling the operation of the public cloud 12) may collude with unauthorised users 18 in an attempt to decrypt the ciphertext CT .
 - The key generation centre 20 is normally operated by or under the control of the data
 25 owners 16 or of trusted third parties. Thus the key generation centre 20 is trusted to generate and update all the cryptographic keys for distribution to the public cloud 12, the management server 22 and the users 18.
 - The management server 22 is normally operated by or under the control of the data
 30 owners 16 or of trusted third parties and thus will not collude with unauthorised users.
- It will be appreciated that the management server 22 cannot decrypt the

ciphertext CT , or the intermediate decrypted cipher text $IDCT$ without, respectively, colluding with either the public cloud 12 or the users 18.

Due to the intermediate step of having the management server conduct an intermediate secondary decryption on the encrypted data 14, it is not possible for a revoked user 18 to
 5 decrypt the encrypted data 14 using only the public key PK and their revoked secret decryption key SDK .

The above general description will now be further exemplified with reference to an embodiment.

The technology can be instantiated by modifying any standard ciphertext-policy attribute-
 10 based encryption (CP-ABE) scheme according the descriptions mentioned above. Standard CP-ABE was accepted as a solution to enforce scalable access control of encrypted data in untrusted servers but suffers from high decryption cost and the inability to handle user revocation and attribute update. The following steps are performed:

Operations by the key generation center 20.

15 The key generation center 20 is operated by the data owners 16 or a trusted third party authority. At the system set up, the key generation center 20 runs the following setup algorithm:

- $Setup(\lambda, U)$: The setup algorithm takes as input a security parameter λ and an attribute universe description U . It generates a tuple (p, G, G_T, e) , where G and G_T
 20 are multiplicative groups of prime order p , and $e: G \times G \rightarrow G_T$ is bilinear map such that it has:
 - Bilinearity: $e(g^a, h^b) = e(g, h)^{ab}$ for all $g, h \in G$ and $a, b \in \mathbb{Z}_p^*$
 - Nondegeneracy: $e(g, h) \neq 1$ whenever $(g, h) \neq 1_G$.
 - Computability: efficient computability for any input pair.

The setup algorithm chooses a generator g of G , a hash function F mapping $\{0, 1\}^*$ to G , and random exponents α and a in Z_p . The setup algorithm then outputs:

- Public parameters: $PK = (g, e(g, g)^\alpha, g^a, F)$
- Master key: $MSK = g^\alpha$

5 Whenever a new user joins the system 10 or a user's set of attributes S is updated, the key generation center 20 runs the following key generation algorithm:

- *KeyGen*(PK, MSK, u, S): The key generation algorithm takes as input the public key PK , the master secret key MSK , a user's identifier u , and the user's set of attributes S . The key generation algorithm chooses random t, z_1, z_2 in Z_p^* , and computes $K = g^{\alpha/(z_1 \cdot z_2)} g^{at}$, $L = g^t$ and $K_x = F(x)^t$ for all x in S . The key generation algorithm then outputs the:

- Secret decryption key $SDK_u = z_1$ and sends it to the user having user identifier u over a secure channel.
- Secret management key $SMK_u = z_2$ and sends (u, SMK_u) to the management server 22 over a secure channel.
- Public decryption key $PDK_u = (K, L, \{K_x\} \text{ for } x \text{ in } S)$ and sends (u, S, PDK_u) to the public cloud 12 over an authenticated channel.

Operations by data owners 16

20 When publishing a document M for storage in the public cloud 12, the data owner 16 runs the following encryption algorithm:

- *Encrypt*(PK, M, T): The encryption algorithm takes as input the public key PK , the data owner's document M and an access policy T . The encryption algorithm chooses random s in Z_p and then outputs
- Ciphertext $CT = (C, C', C_1, D_1, \dots, C_l, D_l)$, where $C = M \cdot e(g, g)^{\alpha s}$, $C' = g^s$ and $C_1, D_1, \dots, C_l, D_l$ are computed based on g^a, F, T , and l random numbers in Z_p (Note: l is a positive integer depending on T , e. g., it is the number of leaf

nodes in the access policy T when it is represented as a rooted tree). The encryption algorithm then sends (CT, T) to the public cloud 12 for storage.

Operations by public cloud 12

The public cloud 12 stores the ciphertext CT and its associated access policy T in a storage system. The public cloud maintains a public cloud list $PC-L$ in which every non-revoked user 18 has an entry in the form of (u, S, PDK_u) . Note that the public cloud 12 receives the threefold entry or triplet (u, S, PDK_u) from the key generation center 20 whenever a new user 18 joins the system or an existing user's set of attributes is updated. If a new user is to be added the new triplet (u, S, PDK_u) is added to the public cloud list $PC-L$. If a set of attributes of an existing user is updated, the new triplet (u, S, PDK_u) replaces the existing entry for that user. If the user is revoked from the system, the corresponding triplet entry (u, S, PDK_u) is deleted from the public cloud list $PC-L$.

When the public cloud 12 receives a request from a user to access a ciphertext CT with an access policy T , the public cloud 12 runs the following primary decryption algorithm:

- 15 - $PC-Decrypt(PK, CT, T, u, S, PDK_u)$: The primary decryption algorithm takes as input the public parameters PK , the ciphertext $CT=(C, C', C_1, D_1, \dots, C_l, D_l)$, the access policy T , the user identifier u , the user's set of attributes S and the public decryption key $PDK_u = (K, L, \{K_x\} \text{ for } x \text{ in } S)$. If the user's triplet entry does not exist in the public cloud list $PC-L$, the primary decryption algorithm rejects the request. Otherwise, the primary decryption algorithm tests if S satisfies T ; if it does not, the primary decryption algorithm rejects the request; and if it does, the primary decryption algorithm decrypts the ciphertext CT using the public decryption key PDK_u to obtain $CT' = e(C', K) / (e(\prod_i C_i^{\omega_i}, L) * \prod_i e(D_i^{\omega_i}, K_i)) = e(g, g)^{sa/(z1+z2)}$. The public cloud then sends $(pct' = (C, CT'), u)$ to the management server 22, where i is taken from the index set I of attributes in both S and T , and ω_i is a constant such that if $\{\lambda_i\}$ are valid shares of any secret s according to T , then $\sum_i \omega_i \lambda_i = s$ for all i in I . Previously above the partially decrypted ciphertext pct' was also referenced as $PDCT$.

Operations by management server 22

The management server maintains a management server list $MS-L$ in which every non-revoked user u has an entry in the form of (u, SMK_u) . The management server 22 receives the pair (u, SMK_u) from the key generation server 20 whenever a new user 18 joins the system 10 or the set of attributes of an existing user 18 is updated. If the user 18 is a new user, the new pair (u, SMK_u) is added to the management server list $MS-L$. If the user is an existing user whose set of attributes is updated, the new pair (u, SMK_u) replaces the existing entry for that user. If the user 18 is to be revoked from the system, the corresponding pair entry (u, SMK_u) is deleted from the management server list $MS-L$.

When the management server receives a partially decrypted ciphertext $pct' = (C, CT')$ for a user identifier u from the public cloud 18, it runs the following secondary decryption algorithm:

- $MS-Decrypt(PK, CT', u, SMK_u)$: The secondary decryption algorithm takes as input the public key PK , the partially decrypted ciphertext $pct' = (C=M \cdot e(g,g)^{s\alpha}, CT'=e(g,g)^{s\alpha/(z_1 \cdot z_2)})$, the user identifier u , and the user's secret management key $SMK_u = z_2$. If the user's entry does not exist in the management server list $MS-L$, the secondary decryption algorithm rejects the decryption request. Otherwise, the secondary decryption algorithm decrypts pct' using SMK_u to obtain a further partially decrypted ciphertext $pct'' = (C, CT'' = CT'^{z_2} = e(g,g)^{s\alpha/z_1})$ and sends pct'' to the user 18. Previously above the partially decrypted ciphertext pct'' was also referenced as IDCT.

Operations by users 18

Upon receiving the partially decrypted ciphertext pct'' , the user 18 runs the following final decryption algorithm:

- $User-Decrypt(PK, pct'', SDK_u)$: The final decryption algorithm takes as input the public parameters PK , the partially decrypted ciphertext $pct'' = (C=M \cdot e(g,g)^{s\alpha}, CT''=e(g,g)^{s\alpha/z_1})$, and the user's secret decryption key $SDK_u = z_1$, and outputs the decrypted message $M=C/CT''^{z_1}$.

Note that most of the decryption operations are performed in PC-Decrypt, while both MS-Decrypt and User-Decrypt just need to perform one exponential operation.

It will be appreciated by persons skilled in the art that numerous variations and/or modifications may be made to the invention as shown in the specific embodiments without
5 departing from the spirit or scope of the invention as broadly described. The present embodiments are, therefore, to be considered in all respects as illustrative and not restrictive.

In the claims which follow and in the preceding description of the invention, except where the context requires otherwise due to express language or necessary implication, the
10 word “comprise” or variations such as “comprises” or “comprising” is used in an inclusive sense, i.e. to specify the presence of the stated features but not to preclude the presence or addition of further features in various embodiments of the invention.

CLAIMS

1. A data security system comprising:
 - a public computer server being arranged to store encrypted data as ciphertext;
 - a public database provided on the public computer server being arranged to
 - 5 store a public decryption key for partially decrypting the ciphertext to yield a partially decrypted ciphertext, to store an attribute set arranged to regulate use of the public decryption key, and to store a user identifier to be associated with a user;
 - a private computer server;
 - a private database provided on the private computer server being arranged to
 - 10 store the user identifier, and to store a first private decryption key for partially decrypting the partially decrypted ciphertext to yield an intermediate decrypted ciphertext;
 - a second private decryption key arranged to be provided to the user for fully decrypting the intermediate decrypted ciphertext to yield unencrypted data; and
 - 15 wherein the respective decrypting of the ciphertext and the partially decrypted ciphertext is only permitted if the user identifier is stored in the public database and in the private database.
2. A data security system as claimed in claim 1, which comprises a key generation
- 20 program being arranged to generate cryptographic keys including the public decryption key, the first private decryption key, and the second private decryption key.
3. A data security system as claimed in claim 2, in which the key generation program is
- 25 arranged to simultaneously amend the public decryption key and user identifier in the public database, and to amend the first private decryption key and user identifier in the private database.
4. A data security system as claimed in claim 2 or 3, in which the key generation
- program is arranged to simultaneously delete the public decryption key and user
- identifier in the public database, and to delete the first private decryption key and
- user identifier in the private database.

5. A data security system as claimed in any one of claims 1 to 4, in which the second private decryption key is unable to decrypt the ciphertext or to decrypt the partially decrypted ciphertext.
6. A data security system as claimed in any one of claims 1 to 5, wherein the public computer server is a public cloud server, an unsecured publically accessible personal computer or an unsecured publically accessible mobile device.
7. A method for operating a data security system comprising the steps of:
 - providing encrypted data as ciphertext on a public computer server;
 - providing a public decryption key to partially decrypt the ciphertext on the public computer server to yield a partially decrypted ciphertext, wherein the public decryption key is associated with an attribute set arranged to regulate use of the public decryption key;
 - transmitting the partially decrypted ciphertext to a private computer server having a first private decryption key to further partially decrypt the partially decrypted ciphertext to yield an intermediate decrypted ciphertext; and
 - transmitting the intermediate decrypted ciphertext to a user having a second private decryption key to fully decrypt the intermediate decrypted ciphertext to yield unencrypted data;
 - wherein the public decryption key and the first private decryption key are associated with a user identifier of the user; and
 - wherein the steps of partially decrypting the ciphertext and partially decrypting the partially decrypted ciphertext are only performed if the user identifier is associated with both the public decryption key and the first private decryption key.
8. A method as claimed in claim 7, wherein the public computer server is a public cloud server, an unsecured publically accessible personal computer or an unsecured publically accessible mobile device.
9. A method as claimed in claim 7 or 8, wherein the step of partially decrypting the ciphertext to yield a partially decrypted ciphertext, comprises decrypting the ciphertext to a substantial extent whereby each of the subsequent steps of decrypting the partially decrypted ciphertext and decrypting the intermediate decrypted ciphertext, respectively comprise performing only minimal computations.

10. A method as claimed in claim 9, in which the minimal computations comprise performing only one exponential operation.
11. A method as claimed in any one of claims 7 to 10, which comprises the step of amending the public decryption key and user identifier on the public computer server, and amending the first private decryption key and user identifier on the private computer server.
12. A method as claimed in any one of claims 7 to 11, which comprises the step of deleting the public decryption key and user identifier from the public computer server, and deleting the first private decryption key and user identifier from the private computer server.