On Using Encryption Techniques to Enhance Sticky Policies Enforcement

Qiang Tang

DIES, Faculty of EEMCS
University of Twente, the Netherlands
q.tang@utwente.nl

Abstract. How to enforce privacy policies to protect sensitive personal data has become an urgent research topic for security researchers, as very little has been done in this field apart from some ad hoc research efforts. The sticky policy paradigm, proposed by Karjoth, Schunter, and Waidner, provides very useful inspiration on how we can protect sensitive personal data, but the enforcement is very weak. In this paper we provide an overview of the state of the art in enforcing sticky policies, especially the concept of sticky policy enforcement using encryption techniques including Public-Key Encryption (PKE), Identity-Based Encryption (IBE), Attribute-Based Encryption (ABE), and Proxy Re-Encryption (PRE). We provide detailed comparison results on the (dis)advantages of these enforcement mechanisms. As a result of the analysis, we provide a general framework for enhancing sticky policy enforcement using Type-based PRE (TPRE), which is an extension of general PRE.

1 Introduction

With the proliferation of sensitive personal information (such as Personal Health Record (PHR) [29]), the related privacy concerns have been the focus of the information security community. A number of international or national legislations, such as the Directive on privacy and electronic communications [11] the Health Insurance Portability and Accountability Act (HIPAA) [31], require that the owner of sensitive data should be able to specify the access control policies.

In practice, usually complex privacy policies are needed to protect sensitive personal data and many entities may get involved [20, 22]. Even managing to specify the policies, without effective enforcement mechanisms, data owners would lose control over their personal information after the initial disclosure. Little has been done so far to directly involve data owners (or entities acting on their behalf) in the enforcement of privacy policies, especially in those sophisticated environments such as healthcare. Take PHR as an example, where a patient monitors several vital functions as part of a disease management program. The attending physician needs to interpret the data, but meanwhile the insurance company or health management organization needs statistical data. The patient may grant full access to the data to physicians, but may want to

www.manaraa.com

limit access for non-clinical personnel. In reality, entities (say, a hospital), acting on the patient's behalf, are incapable of appropriately controlling the confidential information on behalf of their customers. Presently, there does not exist an effective policy enforcement mechanism which allows the patient to enforce privacy policies on its PHR.

Hence, there is an urgent need to have an effective policy enforcement mechanism for data owners to grant their consents on how to use their data and strictly enforce these policies. Karjoth, Schunter, and Waidner [17] introduce the sticky policy paradigm and mechanisms for enterprise privacy enforcement, i.e. a platform for enterprise privacy practices (E-P3P). The concept of sticky policy is to attach privacy policies to data owners' data and drive access control decisions and policy enforcement. This paradigm provides very useful inspiration on how we can protect sensitive personal data. However, the implementation of sticky policy in [17] is rather weak in the sense that the enforcement is not guaranteed and the prevention of modification of the policies is also not guaranteed. Since the proposal of this concept, several following works, as described in Section 2, have considered using encryption techniques to enhance sticky policy enforcement, however, a comprehensive study is required to evaluate these methods.

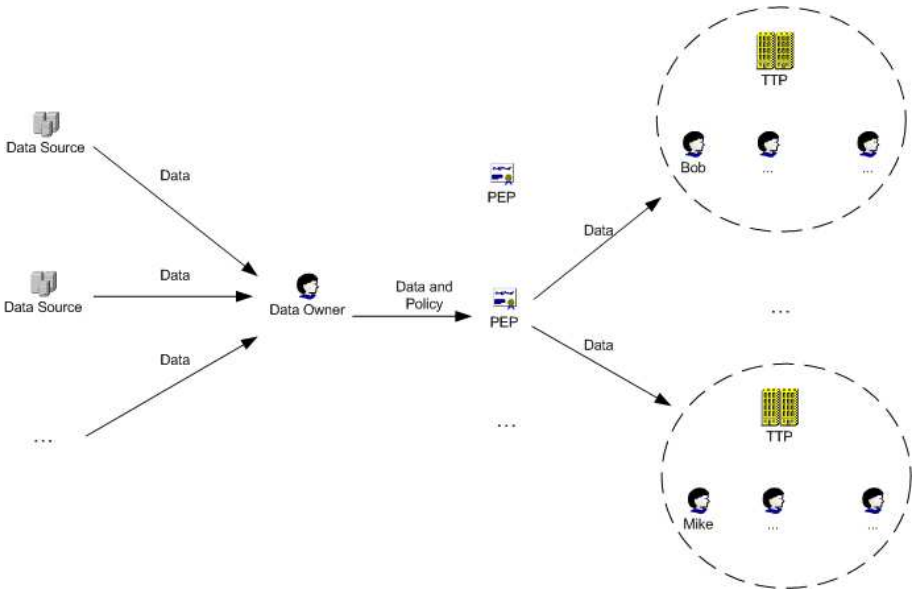

Fig. 1. System Structure for Sticky Policy Enforcement

www.manaraa.com

Contribution. In this paper we evaluate various sticky policy enforcement mechanisms constructed from encryption techniques, including traditional Public-Key Encryption (PKE)[4][1], Identity-Based Encryption (IBE)[27], Attribute-Based Encryption (ABE)[26], and Proxy Re-Encryption (PRE) [7, 18]. A general system structure, as shown in Figure 1, is assumed for the implementation of sticky policy enforcement mechanisms. To our knowledge, this structure generalizes all the existing schemes in the literature. It is worth noting that, for the simplicity reason, Policy Enforcement Point (PEP) represents also other components (such as Policy Decision Point and Policy Information Point) that are standard in implementing access control policies [23].

Detailed comparison results (as shown in Section 4) are obtained by considering the expressiveness of privacy policies, the complexity of updating privacy policies and private keys in the system, and the requirements on the the Trusted Third Party (TTP) and the PEP. In particular, our results show that there is always a tradeoff between the expressiveness and other aspects. For example, the enforcement mechanism using IBE enables more flexible policy expressiveness than the enforcement mechanism using PKE, however, the management of private keys in the former are more complex. Nevertheless, we observe that the enforcement mechanism using PRE provides a general framework to enhance sticky policy enforcement using other encryption techniques in practice. By instantiating the encryption schemes for the delegator and the delegatee in PRE, this enforcement mechanism can be realized in a very flexible way.

In reality, the data owner may want to choose different PEPs to help her enforce her privacy policies on her data at different sensitive levels. For example, she may choose to pay a certain amount money to a trustworthy PEP in order to protect extremely sensitive data, while choose to pay less money to a less trustworthy PEP to protect less sensitive data. In this paper, we propose a sticky policy enforcement mechanism which uses Type-based PRE (TPRE) [30]. This new enforcement mechanism provide the data owner the flexibility to choose different proxies to enforce policies on data at different sensitive levels. Compared with that using PRE, this new enforcement mechanism preserves all the advantages while requiring only one public/private key pair instead of multiple ones. In addition, the new mechanism is more robust against proxy compromises.

Organization. The rest of the paper is organized as follows. In Section 2 we briefly review the literature about sticky policy enforcement. In Section 3 we propose a system structure for sticky policy enforcement and enumerate some of the issues we need to consider in evaluating different enforcement mechanisms. In Section 4 we present the general sticky policy enforcement mechanisms using various encryption techniques (PKE, IBE, ABE, and PRE) and show the comparison results. In Section 5 we provide a general framework to enhance sticky policy enforcement using TPRE. In Section 6 we conclude the paper.

[1] Clearly, IBE and ABE are also public key encryption schemes. Nevertheless, we use the notation PKE to refer the traditional public key encryption schemes [4]. The meanings of various notations should be clear in the contexts.

www.manaraa.com

2 Related Work

The seminal work towards a fine-grained access control framework for sensitive information is proposed by Karjoth, Schunter, and Waidner [17]. They introduce the sticky policy paradigm and mechanisms for enterprise privacy enforcement, i.e. platform for enterprise privacy practices (E-P3P). The concept of sticky policy is the following: when disclose its data, a data owner expresses its consents into privacy policies along with selected opt-in and opt-out choices. Later on, the specified policies should be attached to her data wherever the data moves and drive authorization decisions. In [17], the enforcement of sticky policies relies on trustworthy policy engines integrated with traditional authentication and access control components. Therefore, the data owner needs to trust all possible enterprises when disclosing their data. With this kind of enforcement, leakage of personal information is unavoidable if something goes wrong with the involved entities.

To guarantee only legitimate recipients have access to the sensitive data in a distributed open environment, encryption is an indispensable tool. Intuitively, if the data owner can express her policies into the encryption key, then she can be assured that only the targeted recipients could obtain the data. Note that, public key encryption is more useful in this case because symmetric encryption requires a shared secret key to be pre-distributed. Though, in order to improve efficiency public key encryption and symmetric encryption are always combined using the KEM-DEM paradigm [10], where the public key encryption is used to distribute the symmetric key which will be used to encrypt data. For simplicity, we omit this trick in our discussion and this will not affect our results.

IBE [8, 27] in general has the capability of enabling the encrypter to express her policies for the encrypted data because she can choose any string as the public key. Mont, Pearson, and Bramhall [21] describe a solution to enforce sticky policy by using IBE [8, 27]. In their solution, a sticky policy is mapped to an IBE encryption key which describes the subject and the access condition. Smart [28] introduces similar concepts.

Sahai and Waters [26] propose the concept of Attribute-Based Encryption (ABE), where message senders provide a predicate describing how to share the data. Goyal *et al.* [12] further develop the concept of ABE and propose two different forms, namely Key-Policy ABE and Ciphertext-Policy ABE. In KP-ABE, attributes are used to annotate the ciphertexts and predicates over these attributes are ascribed to users' secret keys, while in CP-ABE attributes are used to describe the users credentials and the predicate over these credentials are attached to the ciphertext by the encrypting party. The authors in [5] propose new schemes for CP-ABE. Clearly, in order to enable the encrypter to express her policies into the public key, the CP-ABE is more suitable than the KP-ABE.

PRE, proposed in [7, 19], is a cryptographic primitive developed to delegate the decryption right from one party (the delegator) to another (the delegatee). Recently, proxy re-encryption has been shown very useful in a number of applications such as access control in file storage [2], email forwarding [32], and law

www.manaraa.com

enforcement [16]. Recently, Tang [30] extends the notion of PRE into TPRE, which allows the delegator to use only a single public/private key pair but be able to restrict the proxy to re-encrypt only a subset of her ciphertexts. Ibraimi *et al.* [15] investigate a similar concept to TPRE to enforce personalized privacy policies.

3 System Structure for Sticky Policy Enforcement

In this paper, we are not supposed to discuss how encryption techniques can (not) be used to build a comprehensive sticky policy enforcement mechanism that solve all issues. Instead, we focus on the confidentiality of personal data and consider a simplified situation: the data owner has some sensitive data that she wants to share with users from an organization, where she may or may not be a member. We assume that an authorization rule is of the form

$$(subject, condition, object),$$

specifying that *subject* can read *object* under the condition *condition*, which is a group of predicates. We further assume a defaulted denial rule, i.e. if there is no explicit policy the request will be denied.

As shown in Figure 1, we assume the following types of components in a sticky policy enforcement system:

- Data owner: She owns data items m_i $(1 \leq i \leq n)$ where n is an integer, and wants to specify and enforce privacy policies over her data.
- PEP: The PEP is trusted by the data owner to store her (encrypted) data and enforce (part of) her privacy policies on her data.
- Trusted Third Party (TTP): The registered users at the TTP are the potential recipient of the data owner's data. We assume that the data owner fully trusts the TTP.

According to the proposed system structure, the disclosure of personal sensitive data works as follows: (1) For each data item m_i, the data owner generates the authorization rule and sends (a transformed version of) m_i and the authorization rule to the PEP. (2) Whenever m_i is requested, the PEP should verify that the authorization is satisfied and sends the ciphertext to the requester if the verification is ok.

In the framework of E-P3P [17], m_i is not transformed, and every entity, which holds the data owner's data, acts as a PEP because it is supposed to enforce the policies attached to the data. Potentially, all PEPs have access to the data owner's data since they are fully responsible for the enforcement and the data is not encrypted. This vulnerability will be mitigated if an enforcement mechanism using encryption technique is adopted, because the data will be transformed into the receipients' ciphertext as shown in Section 4.

www.manaraa.com

In order to compare different enforcement mechanisms, we will take into account the following issues.

1. Expressiveness: Role-Base Access Control (RBAC) [1] and Attribute-Based Access Control (ABAC) [9] are believed to be more expressive than Mandatory Access Control (MAC) [3] and Discretionary Access Control (DAC) [13], and they are also widely used in practice. Preferably, an enforcement mechanism should be efficient in supporting RBAC and ABAC.
2. Requirements on the TTP: The TTP plays an important role in bootstrapping the trust relationships between the data owner and the data receivers by issuing/certifying private/public keys. Preferably, the TTP is only required to be online in the initialization phase and the policy enforcement is transparent to TTP.
3. Requirements on the PEP: Under our assumption, the PEP may be required to store the data owner's data and enforce her policies when a user requests the data. As the PEP is required to be always online, it may be a central target of attacks. In an enforcement mechanism, preferably the violation to the data owner's policy is minimized even if the PEP is compromised.
4. Policy updating: In practice the data owner may need to update her privacy policies on her data from time to time. Preferably, an enforcement mechanism should be efficient for the data owner to do this.
5. Key updating: If cryptographic techniques are employed, key management is always an important concern. The issues we are concerned are: key pairs needs to be issued and certified, key pairs might expire and need to be updated, and private keys might be compromised. Note that key updating may trigger policy updating. Preferably, an enforcement mechanism should be efficient in key updating.

Note that there is an unavoidable vulnerability even if the data is encrypted: a user Bob who is authorized to access the data owner's data may send the data to another user Charlie who is not allowed to access the data. To mitigate this vulnerability, management or legal measures should be taken, like audit techniques and forensics. As shown by Pöhls [24], cryptographic techniques such as digital signature may be used to enforce these measures. A detailed discussion of this issue is outside the scope of this paper and regarded to be an interesting future work.

4 Comparison between Various Enforcement Mechanisms

To protect her sensitive data, an intuitive solution for the data owner is to have a private database, which contains the data items m_i $(1 \leq i \leq n)$ and the corresponding privacy policies, and enforce the policies by herself. Unfortunately, this method is too complex in practice. Alternatively, the data owner can store

www.manaraa.com

her data and the corresponding policies in a database controlled by the Policy Enforcement Point (PEP), which will then provide the enforcement[2]. Upon receiving a request from Bob, the PEP proceeds as follows:

1. Verify Bob's credential to make sure that he is the claimed subject, and make sure the *condition* holds.
2. If the credential and *condition* are ok, send m to Bob through a secure channel.

Clearly, the policy enforcement fully relies on the assumption that the PEP is capable of verifying the requester's credentials, hence, this solution creates serious security concerns. First, the data owner needs to fully trust the PEP because the latter completely controls the data and the enforcement of access control policies. In practice the data owner may not accept this strong trust assumption. Secondly, the PEP becomes a central target of attacks. Once the PEP is compromised, the data owner's data will be compromised. Note that the PEP may try to encrypt the data in this solution to improve the security level, but the above security concerns remain.

Next, we describe the policy enforcement mechanisms using different encryption techniques and provide the comparison results with respect to the issues described in Section 3.

4.1 Enforcement Mechanism using PKE

The main concept of policy enforcement using PKE is letting the data owner encrypt the message using the receiver's public key, and letting the PEP enforce other constraints. In this case, the subject is identified by the identity of the receiver.

Suppose that the TTP uses a PKE scheme ($\mathsf{Setup}, \mathsf{KeyGen}, \mathsf{Encrypt}, \mathsf{Decrypt}$), such as that in [4]. A formal definition of PKE is given in Appendix A. In the initialization phase, the TTP issues a certificate to certify each user's public key. We further assume a user with identity id_r is issued a key pair (pk_{id_r}, sk_{id_r}), where id_r is certified in pk_{id_r}. The enforcement mechanism works as follows.

1. Policy enforcement by the data owner: If the the data owner wants to grant the subject (identified by id_r) access to m_i under the condition *condition*, she proceeds as follows:
   (a) Validate the receiver's public key pk_{id_r}.

[2] Many existing services, such as Google Health (https://www.google.com/health) and Microsoft HealthVault (http://www.healthvault.com/), adopt this solution. In these solutions, the data owner is supposed to store her PHR at Google or Microsoft and specify the policies on how to distribute her data. Later on, Google or Microsoft is supposed to faithfully enforce the data owner's policies.

www.manaraa.com

(b) If the validation is ok, send $(id_r, condition, c_i)$ to the PEP, where $c_i = \mathsf{Encrypt}(m_i, pk_{id_r})$.

2. Policy enforcement by the PEP: When Bob requests the message m_i, the PEP does the following.

   (a) Validate that *condition* holds.

   (b) If the *condition* is ok, send c_i to Bob.

The expressiveness of this enforcement mechanism is coarse because a PKE certificate is usually issued with respect to the identity of the subject. Clearly, this mechanism supports a DAC model well, while it not suitable if a RBAC or ABAC model is going to be used. The RFC specification [25] defines a profile for the use of X.509 attribute certificates in Internet protocols. With attribute-based certificates the expressiveness can be improved, however, this enforcement mechanism is less powerful than that using IBE or ABE. Note that an attribute certificate binds a number of attributes to a public key and is different from the IBE and ABE, where the attributes are actually the public key of the user.

The TTP should provide the service that can be used by the data owner to check the validity of the receiver's public key. The PEP should be semi-trusted in the following sense. An adversary, who is not a subject in the authorization rule but has obtained the ciphertext, cannot decrypt the ciphertext; however, the subject, who is described in the policy, will be able to decrypt the ciphertext. In the latter case, the policy will be violated if the condition is not satisfied when the data is disclosed.

We consider the following situations where policy updating and key updating may happen.

- To update the constraint *condition* in an authorization rule $(id_r, condition, c_i)$, the data owner just needs to inform the PEP to update the parameter *condition*.
- For any authorization rule $(id_r, condition, c_i)$, if the subject id_r is to be changed to id_j, the data owner needs to generate $c_i^* = \mathsf{Encrypt}(m_i, pk_{id_j})$ and let the PEP replace c_i with c_i^*. Since it is infeasible for the data owner to recover m_i from c_i, the data owner needs to find another way to recover m_i in order to compute c_i^*. For example, the data owner may keep a copy of her data in a secure storage for this purpose.
- If the private key sk_{id_r} is compromised, then all authorization rules associated with id_r should be updated. In more detail, a new key pair (pk'_{id_r}, sk'_{id_r}) should be generated for id_r and, for any authorization rule $(id_r, condition, c_i)$, c_i should be replaced with $c'_i = \mathsf{Encrypt}(m_i, pk'_{id_r})$. Basically, the data owner can adopt any possible way as in the previous case. In addition, the data owner may be able to get a copy of the compromised private key sk_{id_r} to recover m_i since this key has been compromised any way.

www.manaraa.com

4.2 Enforcement Mechanism using IBE

The main concept of policy enforcement using IBE is letting the data owner encrypt the message using the receiver's IBE identity, letting the PEP enforce the left constraints, while letting the TTP enforce the policy on issuing the receiver's private key. Different from the case of PKE, an IBE identity could be any string. For example, it can be a normal identity as in the case of PKE concatenated with some other constraints, say "Bob||9:00am - 6:00pm". Furthermore, the identity is also the public key for encryption.

Suppose that the TTP uses an IBE scheme ($\mathsf{Setup}, \mathsf{Keygen}, \mathsf{Encrypt}, \mathsf{Decrypt}$), such as that in [8]. A formal definition of IBE is given in Appendix B. In an IBE scheme, each user can ask the TTP to issue a private key for a claimed identity. In more detail, the enforcement works as follows.

1. Policy enforcement by the data owner: If the data owner wants to grant the subject (identified by id_r) access to m_i under the condition *condition*, she proceeds as follows:
   (a) Validate the public key of the TTP.
   (b) If the validation is ok, send $(id_r, condition, c_i)$ to the PEP, where $c_i = \mathsf{Encrypt}(m_i, id_r)$.
2. Policy enforcement by the PEP: When Bob requests the message m_i, the PEP does the following.
   (a) Validate that *condition* holds.
   (b) If the validation is ok, send (id_r, c_i) to Bob. Note that id_r should be sent to Bob because it will be used by Bob to request his private key from the TTP.
3. Policy enforcement by the TTP: When Bob requests his private key according to the identity id_r, the TTP does the following.
   (a) Validate the identity id_r for Bob.
   (b) If the request is valid, send sk_{id_r} to Bob.

The expressiveness of this enforcement mechanism is finer-grained than that using PKE, because the data owner can embed a normal identity and some constraints into the receiver's IBE identity. The ultimate granularity of privacy policy can be controlled by the construction of id_r. The more constraints in id_r, the finer for the data owner to enforce her policies. On the other hand, however, the more constraints in id_r, the more complexity for the TTP to issue the private key because the TTP needs to check the validity of all the constraints in id_r.

The TTP should be on-line to allow the receiver to request his private key at any time. Simultaneously, the TTP should provide the identity provisioning service so that the data owner can find out the appropriate identity for the potential recipient of her data. Certainly, the TTP also plays the role of a policy enforcement party, given that some constraints are embedded in id_r. In practice,

www.manaraa.com

we may find a tradeoff between the policy granularity for the data owner and the workload for the TTP by balancing the constraints in id_r and *condition*. The PEP should be semi-trusted in the same sense as in the enforcement mechanism using PKE. Note that, in this case, the trust on the PEP can be relaxed since the PEP is required to enforce less constraints than in the case using PKE.

We consider the following situations where policy updating and key updating may happen.

- To update the constraint *condition* in an authorization rule $(id_r, condition, c_i)$, the data owner just needs to inform the PEP to update the parameter *condition*.
- For any authorization rule $(id_r, condition, c_i)$, if the subject id_r is to be changed to id_j, the data owner needs to generate $c_i^* = \mathsf{Encrypt}(m_i, id_j)$ and let the PEP replace c_i with c_i^*. Similar to the case in the enforcement mechanism using PKE, the data owner needs to find another way to recover m_i in order to compute c_i^*.
- If the private key sk_{id_r} is compromised, then all authorization rules associated with id_r should be updated. In more details, a new key pair $(id'_r, sk_{id'_r})$ should be generated and, every authorization rule $(id_r, condition, c_i)$, c_i should be replaced with $(id'_r, condition, c'_i)$ where $c'_i = \mathsf{Encrypt}(m_i, id'_r)$. Basically, the data owner can adopt any possible way as in the previous case. In addition, the data owner may be able to get a copy of the compromised private key sk_{id_r} to recover m_i since this key has been compromised any way.

4.3 Enforcement Mechanism using CP-ABE

The main concept of policy enforcement using CP-ABE is letting the data owner encrypt the message based on an access structure τ which is defined using the attributes in the system, and letting the PEP enforce other constraints.

Suppose the TTP uses a CP-ABE scheme $(\mathsf{Setup}, \mathsf{KeyGen}, \mathsf{Encrypt}, \mathsf{Decrypt})$, such as that in [6]. A formal definition of CP-ABE is given in Appendix C. In the initialization phase, each registered user is issued the private keys associated with the the attributes he has; the TTP publishes the system public key pk. We skip the issue on how to identify the attributes for each user. In more detail, the enforcement works as follows.

1. Policy enforcement by the data owner: If the data owner wants to grant the subject (whose attributes satisfy the access structure τ) access to m_i under the condition *condition*, she proceeds as follows:
   (a) Validate the public key pk of the TTP.
   (b) If the validation is ok, generate $c_i = \mathsf{Encrypt}(m_i, \tau, pk)$ and send the $(\tau, condition, c_i)$ to the PEP.

www.manaraa.com

2. Policy enforcement by the PEP: When Bob requests the message m_i, the PEP does the following.
   (a) Validate that *condition* holds.
   (b) If the *condition* is ok, send (τ, c_i) to Bob.

Due to the flexible expressiveness of CP-ABE, this enforcement mechanism is finer-grained than those using the PKE and IBE in the sense that it is possible to express privacy policy like "if Bob possesses 2 of 10 attributes can read the message m". On the other hand, this enforcement mechanism is less expressive than that using IBE because all attributes are pre-defined by the TTP and the data owner cannot generate new attributes on the fly.

The TTP does not need to be on-line as long as it issues the private keys for each user. As in the case of using IBE, the TTP should provide information about all available attributes so that the data owner can define the access structure for the potential recipient of her data. The PEP should be semi-trusted in the following sense. An adversary, whose attributes do not satisfy the access structure τ, cannot decrypt the ciphertext even if it has obtained the ciphertext; however, other subjects will be able to decrypt the ciphertext. In the latter case, the privacy policy will be violated if the condition is not satisfied when the data is disclosed.

We consider the following situations where policy updating and key updating may happen.

- To update the constraint *condition* in an authorization rule $(\tau, condition, c_i)$, the data owner just needs to inform the PEP to update the parameter *condition*.
- For any authorization rule $(\tau, condition, c_i)$, consider the situation that the τ is to be changed to τ'. In this case, the data owner needs to generate $c_i^* = \mathsf{Encrypt}(m_i, \tau', pk)$ and let the PEP replace c_i with c_i^*. Similar to the case in the enforcement mechanism using PKE, the data owner needs to find another way to recover m_i in order to compute c_i^*.
- For CP-ABE, private key updating is a complex issue. According to existing schemes, in order to revoke some attributes possessed by a user, then the TTP potentially needs to update these attributes in all relevant users. With respect to an enforcement mechanism using CP-ABE, if the private keys associated with τ are compromised, then all authorization rules associated with any attribute in τ may need to be updated.

4.4 Enforcement Mechanism using PRE

PRE provides us a general framework to incorporate other encryption techniques, as the delegator's encryption scheme and the delegatee's encryption scheme could be any of {PKE, IBE}[3]. The main concept of policy enforcement

[3] There is no literature on whether or not CP-ABE can be put in the framework of PRE.

www.manaraa.com

using PRE is letting the data owner (acting as the the delegator in the PRE framework) encrypt the message using her own public key pk_a and partially enforce her policy by assigning a re-encryption key to the PEP, while letting the PEP (acting as the the proxy in the PRE framework) enforce other constraints.

Suppose the data owner uses a scheme $(\mathsf{Setup}_1, \mathsf{KeyGen}_1, \mathsf{Encrypt}_1, \mathsf{Decrypt}_1)$, and the TTP uses a scheme $(\mathsf{Setup}_2, \mathsf{KeyGen}_2, \mathsf{Encrypt}_2, \mathsf{Decrypt}_2)$. A formal definition of PRE is given in Appendix D. In the initialization phase, the data owner obtains a public/private key pair (pk_a, sk_a). Depending on the encryption scheme, the initialization phase for the TTP's domain is the same as in one of the previous enforcement mechanisms. In more detail, the enforcement works as follows.

1. Policy enforcement by the data owner: If the data owner wants to grant the subject (identified by id_r) access to m_i under the condition *condition*, she proceeds as follows:
   (a) Encrypt m_i using her public key to obtain $c_i = \mathsf{Encrypt}_1(m_i, pk_a)$.
   (b) Validate the receiver's public key and the TTP's public parameter.
   (c) If the validation is ok, send $(id_r, condition, c_i)$ and the proxy re-encryption key $rk_{pk_a \to pk_{id_r}} = \mathsf{Pextract}(pk_a, pk_{id_r}, sk_a)$ to the PEP.
2. Policy enforcement by the PEP: When Bob requests the message m_i, the PEP does the following.
   (a) Validate that *condition* holds.
   (b) If the *condition* is ok, send c_i' to Bob, where
   $$c_i' = \mathsf{Preenc}(c_i, rk_{pk_a \to pk_{id_r}}).$$
3. (Optional.) Policy enforcement by the TTP: When Bob requests his private key according to the identifier id_r, the TTP does the following.
   (a) Validate the identity id_r for Bob.
   (b) If the request is valid, send sk_{id_r} to Bob.

Note that the optional step is required only if the TTP uses an IBE scheme. For this enforcement mechanism, both the expressiveness and TTP's role depend on the TTP's encryption scheme. Here, it is more efficient for the PEP with respect to storing the ciphertexts than in other enforcement mechanisms: the PEP does not need to keep the ciphertexts secret but only needs to keep the re-encryption keys secret, while the PEP needs to keep the ciphertext secret in others. On the other hand, the PEP needs to re-encrypt the data owner's ciphertext during the disclosure, which is less efficient than in other enforcement mechanisms where the PEP just needs to forward the ciphertext.

In the enforcement mechanisms using any of {PKE, IBE, or CP-ABE}, the data owner can be assured that, whatever happens to the PEP, only the specified

www.manaraa.com

subject can have access to the data item though the condition may not be satisfied. But, with an enforcement mechanism using PRE, this fact may not be true because the re-encryption key can re-encrypt all of the data owner's ciphertexts. Consider a situation that the data owner stores only $(id_{r_1}, condition, c_i)$ and $(id_{r_2}, condition, c_j)$ at the PEP. In this case, the PEP potentially can re-encrypt c_i,

$$c_i' = \mathsf{Preenc}(c_i, rk_{pk_a \to pk_{id_{r_2}}}),$$

to make the subject identified by id_{r_2} be able to obtain m_i. Therefore, the data owner should have a higher level trust on the PEP if an enforcement mechanism using PRE is used. In Section 5 we show that replacing the PRE with TPRE will mitigate this problem.

We consider the following situations where policy updating and key updating may happen.

- To update the constraint *condition* in an authorization rule $(id_r, condition, c_i)$, the data owner just needs to inform the PEP to update the parameter *condition*.
- This enforcement mechanism is more efficient than others with respect to updating policies. For any authorization rule $(id_r, condition, c_i)$, consider the situation that the id_r is to be changed to id_j. In this case, the data owner only needs to inform the PEP to replace id_r with id_j in the authorization rule, and possibly issues the re-encryption key $rk_{pk_a \to pk_{id_j}}$ if it does not exist. However, in other enforcement mechanisms, the data owner should encrypt all the plaintexts again using the new public key.
- This enforcement mechanism is more efficient for the PEP than others with respect to updating the receiver's key pair. If the private key sk_{id_r} is expired or compromised, then all authorization rules associated with id_r should be updated. In this case, the data owner only needs to generate a new re-encryption key $rk_{pk_a \to pk'_{id_r}}$ for the PEP. However, for other mechanisms, the data owner needs to recover the relevant plaintexts first and encrypt them again using the receiver's new public key.
- If the data owner's private key pk_a is compromised, then she needs to generate a new key pair (pk'_a, sk'_a) and inform the PEP to replace every re-encryption key $rk_{pk_a \to pk_{id_r}}$ with $rk_{pk'_a \to pk_{id_r}}$. For any policy $(id_r, condition, c_i)$, the data owner needs to generate $c_i' = \mathsf{Encrypt}(m_i, pk'_a)$ and let the PEP replace c_i with c_i'.

Consider the following case where the data owner needs to enforce her policies on the same data item m_i for multiple receivers. With an enforcement mechanism using PRE, the data owner only needs to encrypt m_i once and possibly needs to generate a re-encryption key for each recipient. Note the fact that, if the data owner grants the same recipient multiple data items, she only needs to generate the re-encryption key once. However, it is less efficient in other enforcement mechanisms because the data owner should encrypt m_i using each recipient's

www.manaraa.com

public key. In addition, the data owner may need to enforce different policies on a data item m_i at different time, an enforcement mechanism using PRE does not require her to store a copy of m_i while other enforcement mechanisms do have this requirement.

Consider the following case where the data owner needs to enforce policies on here data which are generated by third parties. With an enforcement mechanism using PRE, the third party can encrypt the messages using the data owner's public key and deposit the ciphertexts at the PEP. Later on, the data owner can avoid the encryption operations in enforcing her policies. However, for other mechanisms, the data owner needs to encrypt all the plaintexts from the third party to enforce her policies. Hence, the PRE mechanism is more efficient than others in these scenarios. In next section we show that this advantage is more obvious if a TPRE is adopted.

5 Enforcement Mechanism using TPRE

In this section we propose an enforcement mechanism using TPRE in order to mitigate the problems with that using PRE.

5.1 The Motivation

With respect to the traditional PRE schemes (e.g. [7, 19]) and their applications, we observe the following drawbacks.

- With a re-encryption key, the proxy is capable of re-encrypting all the ciphertexts of the delegator, so that any delegatee potentially can obtain all of the delegator's data. In order to revoke a compromised re-encryption key, the delegator's key pair should be revoked.
- The delegator should trust the proxy to properly re-encrypt all her ciphertexts. If the delegator has a number of data sets which has different sensitive levels, the delegator needs to choose a different key pair for each possible subset of his messages and choose a proxy to delegate her decryption right. In practice, this approach is not very realistic because the delegator needs to maintain a number of key pairs.

As shown in Figure 2, with a TPRE, the delegator can generate a re-encryption key for each combination of receiver and message type, and every re-encryption key only works for the intended type of message. A formal definition of TPRE is in Appendix D.

With TPRE, the problems associated with PRE are eliminated. The delegator can categorize her data into different subsets and delegate the decryption

www.manaraa.com

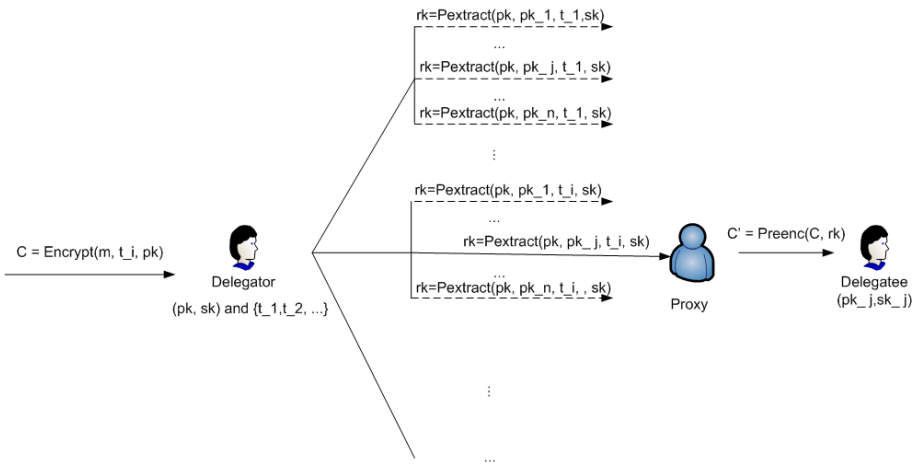

Fig. 2. Overview of TPRE

right of each subset through a different proxy, while the delegator only needs one key pair to do so. As a result, the delegator can choose a different proxy to delegate the re-encryption right based on the sensitive level of her data. Compromise of one re-encryption key will only affect one type of messages. In order to revoke a compromised re-encryption key of type t, the delegator can just generate a new re-encryption key for a new type t^* which could be defined to contain the same messages of type t.

5.2 Enforcement Mechanism using TPRE

The main concept of enforcing sticky policy using TPRE is the same as in the case of using PRE. However, in practice, the data owner only needs to have one key pair but can choose different proxies based on her trust and the sensitive level of her data. Suppose the data owner categorizes her data into types $\{t_1, t_2, \cdots\}$ and uses a type-based public key encryption scheme $(\mathsf{Setup}_1, \mathsf{KeyGen}_1, \mathsf{Encrypt}_1, \mathsf{Decrypt}_1)$, as defined in Appendix A. Suppose the TTP uses an encryption scheme $(\mathsf{Setup}_2, \mathsf{KeyGen}_2, \mathsf{Encrypt}_2, \mathsf{Decrypt}_2)$. In the initialization phase, the data owner obtains a public/private key pair (pk_a, sk_a) and publishes her categorization of data. In more detail, the enforcement works as follows.

1. Policy enforcement by the data owner: If the data owner wants to grant the subject (identified by id_r) access to a message m_i of type t_i under the condition *condition*, she proceeds as follows:
   (a) Encrypt m_i using her public key to obtain $c = \mathsf{Encrypt}_1(m_i, t_i, pk_a)$.
   (b) Validate the receiver's public key pk_{id_r} and the TTP's public parameter.

www.manaraa.com

(c) If the validation is ok, send $(id_r, t_i, condition, c_i)$ and the re-encryption key $rk_{pk_a \xrightarrow{t_i} pk_{id_r}}$ to the PEP, where

$$rk_{pk_a \xrightarrow{t_i} pk_{id_r}} = \mathsf{Pextract}(pk_a, pk_{id_r}, t_i, sk_a).$$

2. Policy enforcement by the PEP: When Bob requests the message m_i, the PEP does the following.

   (a) Validate that *condition* holds.

   (b) If the *condition* is ok, send c_i' to Bob, where

$$c_i' = \mathsf{Preenc}(c_i, t_i, rk_{pk_a \xrightarrow{t_i} pk_{id_r}}).$$

3. (Optional.) Policy enforcement by the TTP: When Bob requests his private key according to the identifier id_r, the TTP does the following.

   (a) Validate the identity id_r for Bob.

   (b) If the request is valid, send sk_{id_r} to Bob.

Note that the optional step is required only if the TTP uses an IBE scheme. This enforcement mechanism possesses all the advantages of the enforcement mechanism using PRE, while relaxes the security requirements on the PEPs.

The data owner can choose different PEPs to enforce her privacy policies on her sensitive data at different sensitive levels. If one PEP, which possesses the re-encryption key $rk_{pk_a \xrightarrow{t_i} pk_{id_r}}$ has been compromised, then other types of data will not be affected (in contrast to that using PRE) given CPA/CCA securities is achieved. To revoke her compromised re-encryption key, the data owner needs to create a new type $t_i^\dagger$ so that all all data in the category t_i should be redefined to be in the category $t_i^\dagger$. Correspondingly, any authorization rule $(id_r, t_i, condition, c_i)$ should be replaced with $(id_r, t_i^\dagger, condition, c_i^\dagger)$, where

$$c_i^\dagger = \mathsf{Encrypt}_1(m_i, t_i^\dagger, pk_a).$$

When the data owner's sensitive data are generated by a third party (as in the cases of Google Health and Microsoft HealthVault), this enforcement mechanism provides more flexibility. With careful categorization, the data owner can define her privacy policies for each type of her data in the form of $(id_r, t_i, condition)$. Later on, any third party can just send the ciphertext together with the type information to the PEP, which will then be able to connect it with the corresponding privacy policies. In this aspect, this enforcement mechanism is better than that using PRE.

www.manaraa.com

6 Conclusion

In this paper we have provided an overview of using various public key encryption techniques to enforce sticky policies. The comparison results indicate that the enforcement mechanism using PRE provides a better performance in the aspects of the policy updating and key updating for the data receivers. The enforcement mechanism using TPRE further improves the performance by mitigating the key management inefficiency for the data owner. However, we should note that what we have done is only a step towards a comprehensive sticky policy enforcement mechanism. A number of issues are still open to further research.

1. In the the TPRE framework proposed by Tang [30], the encryption schemes have only been instantiated to PKE. Therefore, it is interesting to investigate the instantiations using IBE or even CP-ABE.
2. In our discussions, we have assumed that the data owner will define all the policies for her data, which means all authorizations are directly from the data owner. In practice, we may want a more smooth process: If Bob has obtained the data item m_i in the form of c_i' (a ciphertext of his public key), if Bob is authorized to re-distribute m_i to another user Mike, then it is more efficient and potentially secure for Bob to re-encrypt c_i' into c_i'', which is a ciphertext under Mike's public key. It is interesting to investigate the possibility of extend the TPRE framework to multi-level delegations (as in [14]), and instantiate the encryption schemes to IBE and PKE.
3. Another interesting research direction is to take obligation into account and incorporate the corresponding enforcement techniques into the sticky policy enforcement mechanisms.

References

1. American National Standards Institute (ANSI). ANSI INCITS 359-2004 for Role Based Access Control, 2004.
2. G. Ateniese, K. Fu, M. Green, and S. Hohenberger. Improved proxy re-encryption schemes with applications to secure distributed storage. *ACM Transactions on Information and System Security*, 9(1):1–30, 2006.
3. D. Bell and L. La Padula. Secure computer systems: A mathematical model. Technical Report Technical report MTR-2547, MITRE Corp., 1973.
4. M. Bellare, A. Desai, D. Pointcheval, and P. Rogaway. Relations among notions of security for public-key encryption schemes. In H. Krawczyk, editor, *Advances in Cryptology — CRYPTO 1998*, volume 1462 of *Lecture Notes in Computer Science*, pages 26–45. Springer, 1998.
5. J. Bethencourt, A. Sahai, and B. Waters. Ciphertext-policy attribute-based encryption. In *2007 IEEE Symposium on Security and Privacy (S&P 2007)*, pages 321–334. IEEE Computer Society, 2007.
6. J. Bethencourt, A. Sahai, and B. Waters. Ciphertext-Policy Attribute-Based Encryption. pages 321–334. IEEE Computer Society, 2007.

www.manaraa.com

7. M. Blaze, G. Bleumer, and M. Strauss. Divertible protocols and atomic proxy cryptography. In K. Nyberg, editor, *Advances in Cryptology - EUROCRYPT '98, International Conference on the Theory and Application of Cryptographic Techniques*, volume 1403 of *Lecture Notes in Computer Science*, pages 127–144. Springer, 1998.
8. D. Boneh and M. K. Franklin. Identity-based encryption from the weil pairing. In J. Kilian, editor, *Advances in Cryptology - CRYPTO 2001, 21st Annual International Cryptology Conference*, volume 2139 of *Lecture Notes in Computer Science*, pages 213–229. Springer, 2001.
9. M. J. Covington and M. R. Sastry. A contextual attribute-based access control model. In R. Meersman, Z. Tari, and P. Herrero, editors, *On the Move to Meaningful Internet Systems 2006: OTM 2006 Workshops*, volume 4278 of *Lecture Notes in Computer Science*, pages 1996–2006. Springer, 2006.
10. R. Cramer and V. Shoup. Design and analysis of practical public-key encryption schemes secure against adaptive chosen ciphertext attack. *SIAM Journal on Computing*, 33(1):167–226, 2004.
11. Council European Parliament. Directive 2002/58/EC of the European parliament and of the council of 12 july 2002 concerning the processing of personal data and the protection of privacy in the electronic communications sector (directive on privacy and electronic communications). *Official Journal*, L201:37–47, 2002.
12. V. Goyal, O. Pandey, A. Sahai, and B. Waters. Attribute-based encryption for fine-grained access control of encrypted data. In A. Juels, R. N. Wright, and S. De Capitani di Vimercati, editors, *Proceedings of the 13th ACM Conference on Computer and Communications Security, CCS 2006*, pages 89–98. ACM, 2006.
13. G. Graham and P. Denning. Protection – principles and practice. In *Proceedings of the Spring Jt. Computer Conference*, volume 40, pages 417–429, 1972.
14. M. Green and G. Ateniese. Identity-based proxy re-encryption. In J. Katz and M. Yung, editors, *Applied Cryptography and Network Security, 5th International Conference*, volume 4521 of *Lecture Notes in Computer Science*, pages 288–306. Springer, 2007.
15. L. Ibraimi, Q. Tang, P. Hartel, and W. Jonker. A type-and-identity-based proxy re-encryption scheme and its application in healthcare. In W. Jonker and M. Petkovic, editors, *Secure Data Management, SDM 2008*, volume 5159 of *Lecture Notes in Computer Science*, pages 185–198. Springer, 2008.
16. A. Ivan and Y. Dodis. Proxy cryptography revisited. In *Proceedings of the Network and Distributed System Security Symposium*. The Internet Society, 2003.
17. G. Karjoth, M. Schunter, and M. Waidner. Platform for enterprise privacy practices: Privacy-enabled management of customer data. In R. Dingledine and P. F. Syverson, editors, *Privacy Enhancing Technologies, Second International Workshop*, volume 2482 of *Lecture Notes in Computer Science*, pages 69–84. Springer, 2002.
18. M. Mambo and E. Okamoto. Proxy Cryptosystems: Delegation of the Power to Decrypt Ciphertexts. *IEICE Transactions on Fundamentals of Electronics, Communications and Computer Sciences*, 80(1):54–63, 1997.
19. M. Mambo and E. Okamoto. Proxy cryptosystems: Delegation of the power to decrypt ciphertexts. *IEICE TRANSACTIONS on Fundamentals of Electronics, Communications and Computer Sciences*, E80-A(1):54–63, 1997.
20. A. D. Miller and W. K. Edwards. Give and take: a study of consumer photo-sharing culture and practice. In *CHI '07: Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 347–356. ACM, 2007.

www.manaraa.com

21. M. C. Mont, S. Pearson, and P. Bramhall. Towards accountable management of identity and privacy: Sticky policies and enforceable tracing services. In *14th International Workshop on Database and Expert Systems Applications (DEXA'03)*, pages 377–382. IEEE Computer Society, 2003.
22. J. S. Olson, J. Grudin, and E. Horvitz. A study of preferences for sharing and privacy. In *CHI '05: CHI '05 extended abstracts on Human factors in computing systems*, pages 1985–1988. ACM, 2005.
23. Organization for the Advancement of Structured Information Standards (OASIS). eXtensible Access Control Markup Language (XACML) Version 2.0, 2005.
24. H. C. Pöhls. Verifiable and revocable expression of consent to processing of aggregated personal data. In L. Chen, M. Dermot Ryan, and G. Wang, editors, *Information and Communications Security, 10th International Conference, ICICS 2008*, volume 5308 of *Lecture Notes in Computer Science*, pages 279–293. Springer, 2008.
25. RFC 3281. An Internet Attribute Certificate Profile for Authorization, 2002.
26. A. Sahai and B. Waters. Fuzzy identity-based encryption. In R. Cramer, editor, *Advances in Cryptology — EUROCRYPT 2005*, volume 3494 of *Lecture Notes in Computer Science*, pages 457–473. Springer, 2005.
27. A. Shamir. Identity-based cryptosystems and signature schemes. In *Advances in Cryptology, Proceedings of CRYPTO '84*, volume 196 of *Lecture Notes in Computer Science*, pages 47–53. Springer, 1985.
28. N. P. Smart. Access Control Using Pairing Based Cryptography. In M. Joye, editor, *Topics in Cryptology, CT-RSA 2003*, volume 2612 of *Lecture Notes in Computer Science*, pages 111–121. Springer, 2003.
29. P.C. Tang, J.S. Ash, D.W. Bates, J.M. Overhage, and D.Z. Sands. Personal Health Records: Definitions, Benefits, and Strategies for Overcoming Barriers to Adoption. *Journal of the American Medical Informatics Association*, 13(2):121–126, 2006.
30. Q. Tang. Type-based proxy re-encryption and its construction. In D. R. Chowdhury and V. Rijmen, editors, *Proceeding of the 9th International Conference on Cryptology in India (INDOCRYPT 2008)*, volume 5365 of *Lecture Notes in Computer Science*, pages 130–144. Springer, 2008.
31. The US Department of Health and Human Services. Summary of the HIPAA Privacy Rule, 2003. http://www.hhs.gov/ocr/privacysummary.pdf.
32. L. Wang, Z. Cao, T. Okamoto, Y. Miao, and E. Okamoto. Authorization-Limited Transformation-Free Proxy Cryptosystems and Their Security Analyses*. *IEICE Transactions on Fundamentals of Electronics, Communications and Computer Sciences*, (1):106–114, 2006.

Appendix A: Algorithm Definitions for PKE

A PKE scheme [4] involves a Trusted Third Party (TTP) and users, and consists of four algorithms ($\mathsf{Setup}, \mathsf{KeyGen}, \mathsf{Encrypt}, \mathsf{Decrypt}$) which are defined as follows.

- $\mathsf{Setup}(k)$: Run by the TTP, this algorithm takes a security parameter k as input and generates the public parameter $params$, which is an implicit input to other algorithms.
- $\mathsf{KeyGen}(k)$: Run by a user, this algorithm generates a key pair (pk, sk).

www.manaraa.com

– $\mathsf{Encrypt}(m, pk)$: Run by the message sender, this algorithm takes a message m and a public key pk as input, and outputs a ciphertext c encrypted under the public key pk.
– $\mathsf{Decrypt}(c, sk)$: Run by the message receiver, this algorithm takes a ciphertext c and the private key sk as input, and outputs the message m.

Type-based PKE enables a message sender to explicitly include some type information in the encryption process. A type-based PKE consists of four algorithms ($\mathsf{Setup}, \mathsf{KeyGen}, \mathsf{Encrypt}, \mathsf{Decrypt}$), where Setup and KeyGen are defined as above, and

– $\mathsf{Encrypt}(m, t, pk)$: Run by the message sender, this algorithm takes a message m, a type string t, and a public key pk as input, and outputs a ciphertext c encrypted under the public key pk. Note that both c and t should be sent to the message receiver.
– $\mathsf{Decrypt}(c, t, sk)$: Run by the message receiver, this algorithm takes a ciphertext c, a message type t, and the private key sk as input, and outputs the message m.

Note that Type-based IBE can be defined in the same way.

Appendix B: Algorithm Definitions for IBE

An IBE scheme consists of four algorithms ($\mathsf{Setup}, \mathsf{KeyGen}, \mathsf{Encrypt}, \mathsf{Decrypt}$).

– $\mathsf{Setup}(k)$: Run by the TTP, this algorithm takes a security parameter k as input and generates the public parameter is $params$ and a master key mk. The public parameter $params$ is an implicit input to other algorithms.
– $\mathsf{KeyGen}(id, mk)$: Run by the TTP, this algorithm takes an identifier id and the master key mk as input, and outputs the private key sk_{id} corresponding to id.
– $\mathsf{Encrypt}(m, id)$: Run by the message sender, this algorithm takes a message m and an identifier id as input, and outputs a ciphertext c encrypted under the public key corresponding to id.
– $\mathsf{Decrypt}(c, sk_{id})$: Run by the user with identifier id, this algorithm takes a ciphertext c and the private key sk_{id} as input, and outputs the message m.

Appendix C: Algorithm Definitions for CP-ABE

A CP-ABE scheme consist of four algorithms ($\mathsf{Setup}, \mathsf{KeyGen}, \mathsf{Encrypt}, \mathsf{Decrypt}$), defined as follows.

– $\mathsf{Setup}(k)$: This algorithm takes as input a security parameter k and outputs the public parameters pk and a master key mk.

www.manaraa.com

– $\mathsf{KeyGen}(\omega, mk)$. This algorithm takes as input the master key mk and a set of attributes ω, and outputs a private key sk_ω associated with ω.
– $\mathsf{Encrypt}(m, \tau, pk)$. This algorithm takes a message m, an access tree τ, and the receiver's public key pk as input, and outputs a ciphertext c_τ which contains τ. For example, if there are three attributes $\{a_1, a_2, a_3\}$ then τ could be $(a_1 \bigwedge a_2) \bigvee a_3$.
– $\mathsf{Decrypt}(c_\tau, sk)$. This algorithm takes as input a ciphertext c_τ, a secret key sk_ω of private attributes for a set ω, and it outputs a message m or an error symbol $\perp$.

Appendix D: Algorithm Definitions for PRE

Proxy re-encryption is a cryptographic primitive developed to delegate the decryption right from one party (the delegator) to another (the delegatee). Suppose that the delegator with key pair (pk_a, sk_a) is registered at TTP_1 with an encryption scheme

$$(\mathsf{Setup}_1, \mathsf{KeyGen}_1, \mathsf{Encrypt}_1, \mathsf{Decrypt}_1)$$

and the delegatee with key pair (pk_r, sk_r) is registered at TTP_2 with another encryption scheme

$$(\mathsf{Setup}_2, \mathsf{KeyGen}_2, \mathsf{Encrypt}_2, \mathsf{Decrypt}_2).$$

Note that both the encryption schemes could be any of {PKE, IBE}. Apart from the above algorithms, a PRE scheme consists of the following two new algorithms:

– $\mathsf{Pextract}(pk_a, pk_r, sk_a)$: This algorithm takes the delegator's public key pk_a, the delegatee's public key pk_r, the delegator's private key sk_a and outputs the proxy key $rk_{pk_a \to pk_r}$ to the proxy.
– $\mathsf{Preenc}(c, rk_{pk_a \to pk_r})$: Run by the proxy, this algorithm takes a ciphertext c for the delegator and the re-encryption key $rk_{pk_a \to pk_r}$ as input, and outputs a new ciphertext c' for the delegatee.

In the case of TPRE, the delegator uses a type-based encryption scheme $(\mathsf{Setup}_1, \mathsf{KeyGen}_1, \mathsf{Encrypt}_1, \mathsf{Decrypt}_1)$ as defined in Appendix A. The algorithms $\mathsf{Pextract}$ and Preenc are defined as follows.

– $\mathsf{Pextract}(pk_a, pk_r, t, sk_a)$: This algorithm takes the delegator's public key pk_a, the delegatee's public key pk_r, a message type t, the delegator's private key sk_a as input and outputs the delegation key $rk_{pk_a \xrightarrow{t} pk_r}$.
– $\mathsf{Preenc}(c, t, rk_{pk_a \xrightarrow{t} pk_r})$: Run by the proxy, this algorithm takes a ciphertext c (for the delegator), a message type t, and the re-encryption key $rk_{pk_a \xrightarrow{t} pk_r}$ as input, and outputs a new ciphertext c'.

www.manaraa.com