Exploring Type-and-Identity-Based Proxy Re-encryption Scheme to Securely Manage Personal Health Records

Luan Ibraimi[*1], Qiang Tang[*2], Pieter Hartel[*3], Willem Jonker[*#4]

[*] *Faculty of EEMCS, University of Twente*

The Netherlands

[1] ibraimi@utwente.nl

[2] q.tang@utwente.nl

[3] pieter.hartel@utwente.nl

[#] *Philips Research Eindhoven*

The Netherlands

[4] willem.jonker@philips.com

Abstract. Commercial Web-based Personal-Health Record (PHR) systems can help patients to share their personal health records (PHRs) anytime from anywhere. PHRs are very sensitive data and an inappropriate disclosure may cause serious problems to an individual. Therefore commercial Web-based PHR systems have to ensure that the patient health data is secured using state-of-the-art mechanisms. In current commercial PHR systems, even though patients have the power to define the access control policy on who can access their data, patients have to trust entirely the access-control manager of the commercial PHR system to properly enforce these policies. Therefore patients hesitate to upload their health data to these systems as the data is processed unencrypted on untrusted platforms.

Recent proposals on enforcing access control policies exploit the use of encryption techniques to enforce access control policies. In such systems, information is stored in an encrypted form by the third party and there is no need for an access control manager. This implies that data remains confidential even if the database maintained by the third party is compromised. In this paper we propose a new encryption technique called a type-and-identity-based proxy re-encryption scheme which is suitable to be used in the healthcare setting. The proposed scheme allows users (patients) to securely store their PHRs on commercial Web-based PHRs, and securely share their PHRs with other users (doctors).

Keywords: Proxy Re-encryption, Identity-Based Encryption, Personal Health Records, Access Control Policy, Privacy

Introduction

Recently, healthcare providers have started to use electronic health record systems which have significant benefits such as reducing healthcare costs, increasing the patient safety, improving the quality of care and empowering patients to more actively manage their health. There are a number of initiatives for adoption of electronic health records (EHRs) from different governments around the world, such as the directive on privacy and electronic communications in the U.S. known as the Health Insurance Portability and Accountability Act (HIPAA) [1], which specify rules and standards to achieve security and privacy of health data. While EHR systems capture health data entered by health care professionals and access to health data is tightly controlled by existing legislations, personal health record (PHR) systems capture health data entered by individuals and stay outside the scope of this legislation. Before going into details on how to address the confidentiality issues, let us introduce the definition of PHR system [4]:

"An electronic application through which individuals can access, manage and share their health information, and that of others for whom they are authorized, in a private, secure, and confidential environment."

PHR systems are unique in their design since they try to solve the problem that comes from scattering of medical information among many healthcare providers which leads to unnecessary paper work and medical mistakes [4]. The PHR contains all kinds of health-related information about an individual (say, Alice) [5]. Firstly, the PHR may contain medical data that Alice has from various medical service providers, for example about surgery, illness, family history, vaccinations, laboratory test results, allergies, drug reactions, etc. Secondly, the PHR may also contain information collected by Alice herself, for example weight change, food statistics, and any other information connected with her health. Controlling access to PHRs is one of the central themes in deploying a secure PHR system. Inappropriate disclosure of the PHRs may cause an individual serious problems. For example, if Alice has some disease and a prospective employer obtains this, then she might be discriminated in finding a job.
Commercial efforts to build Web-based PHR systems, such as Microsoft HealthVault [2] and Google Health [3], allow patients to store and share their PHRs with different healthcare providers. In these systems the patient has full control over her PHRs and plays the role of the security administrator - a patient decides who has the right to access which data. However, the access control model of these applications does not give a patient the flexibility to specify a fine-grained access-control policy. For example, today's Google Health access control is *all-or-nothing* - so if a patient authorizes her doctor to see only one PHR, the doctor will be able to see all other PHRs. Another problem is that the data has to be stored on a central server locked by the access control mechanism provided by Microsoft HealthVault or Google Health, and the patient loses control once the data is sent to the server. PHRs may contain sensitive information such as details of a patients disease, drug usage, sexual preferences, therefore, many patients are worried whether their PHRs will be treated as confidential by companies running data centers. Inappropriate disclosure of a PHR can change patients life, and there may be no way to repair such harm financially or technically. Therefore, it is crucial to protect PHRs when they are uploaded and stored in commercial Web-based systems.

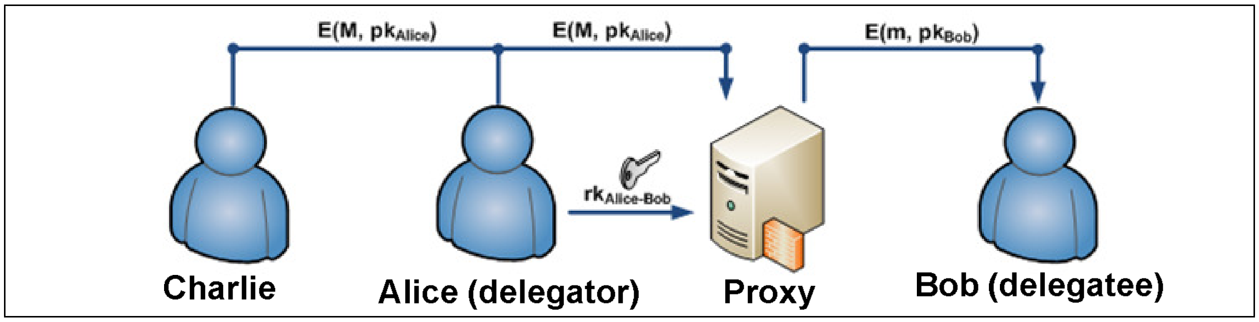

Figure 1: Proxy Re-Encryption

Problem Statement

The problem addressed in this paper is the confidentiality of patient PHRs stored in commercial Web-based PHR systems. A solution to the problem is a system which would have the following security requirements:

- Protect patient PHRs from third parties (from the commercial Web-based PHR systems).
- Provide end-to-end security.
- Allow only authorized users to have access to the patient PHRs.
- Allow the patient to change the access policy dynamically.

Contributions

To solve the identified problem we propose a new public key encryption scheme called type-and-identity-based proxy re-encryption which helps patients to store their PHRs on commercial Web-based PHR systems, and to share their PHRs securely with doctors, family and friends. In public key encryption, each user has a key pair (private/public key) and everyone can encrypt a message using a public key, also referred to as the encryption key, but only users who have the associated private key, also referred to as the decryption key, can decrypt the encrypted message. Proxy re-encryption is a cryptographic method developed to delegate the decryption right from one party, the delegator, to another, the delegatee. In a proxy re-encryption scheme, the delegator assigns a key to a proxy to re-encrypt all messages encrypted with the public key of the delegator such that the re-encrypted ciphertexts can be decrypted with the private key of the delegatee. The Proxy is a semi-trusted entity i.e. it is trusted to perform only the ciphertext re-encryption, without knowing the private keys of the delegator and the delegatee, and without having access to the plain text. In the context of public key encryption, proxy re-encryption is used to forward encrypted messages without revealing the plaintext. For example, in Figure 1, Alice (delegator) is president of a company who wants to allow her secretary, Bob (delegatee), to read encrypted emails from Charlie when Alice is on vacation (Alice cannot give to Bob her private key). Using proxy re-encryption Alice can compute a re-encryption key which would allow the Proxy to transform a ciphertext for Alice generated by Charlie into a ciphertext for Bob, thus, Bob can decrypt the re-encrypted data using his private key. In practice the role of the Proxy can be played by a commercial enterprise which has enough computation power to perform re-encryption services for a large number of users. Our proposed scheme is suitable for the healthcare setting and has the following benefits:

- Allow the patient to store her PHRs in a protected form on semitrusted commercial Web-based PHR system. The commercial Web-based PHR system cannot access the data since the data is stored in an encrypted form.
- Allow the patient to define a fine-grained access control policy. The patient only has to compute the re-encryption key and forward it the Proxy which will re-encrypt the data without decrypting them such that the intended user (e.g.,doctor) can decrypt the re-encrypted data using his private key. In addition to that, the scheme allows the patient to change dynamically the access policy without necessarily decrypting the data.
- Allows the patient to categorize her messages into different types, and delegate the decryption right of each type to the doctor through a Proxy. Data categorization is needed since different data may have different levels of privacy requirements.

Web-based PHRs

In this section we discuss current Web-based PHRs systems such as Microsoft HealthVault and Google Health and their access control mechanisms. Moreover, we discuss existing techniques to enforce access policies using cryptography and discuss their limitations.
In Microsoft HealthVault each patient has an account and an identifier. Each account has a special role called a *custodian* who has the right to view and modify all records, and grant or revoke access to users (e.g.,doctors) and applications. Both users and applications have to be registered with HealthVault in order to access other accounts. An application is a third party service which can read data stored in HealthVault records, store new data, or transfer data from one account to another account. An application can access patient accounts in two ways: a) online access - access the account when the patient is logged on to her account, and b) offline access – access the account at any time. If the patient uses an application for the first time, and the application requires access to a health record, the application sends an access request to the patient. After the patient approves the request, the application can access the health information. HealthVault uses discretionary access control (DAC) [6] where access to a patient PHR is based on user identity (e.g. email). For example, if the patient wants to allow the doctor to see her record, the patient has to send a sharing invitation to the doctors' email and within 3 days the doctor has to accept or reject the invitation. The doctor may have one of the following permissions: a) view patient information, b) view and modify patient information or, c) act as a custodian. Microsoft HealthVault defines 74 types of information and allows granular access control for non-custodians. For example, the patient can allow the doctor to have access only to records of type *Allergy*, and block access to records of type *Family History*. However, the patient does not have the flexibility to grant access to the doctor to an individual record. When the patient grants access to the doctor to her health information, the patient can also specify an expiration date. After the expiration date, the doctor would not be able to access the patient information. However, the patient has the option to remove sharing access at any time (even before the expiration date).
Similar to Microsoft HealthVault, Google Health allows a patient to import her PHRs, add test

results and add information about allergies, among others. In Google Health each patient has an account and an identifier and a user (e.g.,doctor) has to be registered to Google Health to access others health information. If the patient wants to allow the doctor to see her PHRs, the patient has to send an invitation to doctors' email, and within 30 days the doctor has to accept or reject the invitation. A patient can share her PHRs with other users or Google partners including Walgreens, CVS Long Drugs pharmacies, Cleveland Clinic and etc. However, data sharing in Google Health is *all-or-nothing*. The patients does not have the flexibility to chose a fine-grained access policy to share her data. For example, once the patient allows the doctor to see her blood result test, the doctor can access all health information of the patient.
In the access-control mechanisms of Microsoft HealthVault and Google Health the receiving end of the information must provide a set of credentials to the access control manager (ACM) who is responsible for enforcing the access control policies. The ACM checks whether user credentials satisfy the access control policy. If so, the user can read the resource, otherwise not. However, the main limitation of this is that the patient still needs to trust the Microsoft HealthVault and Google Health to enforce access control policies when disclosing data, and specially the access control decisions and privacy enforcement has to be enforced when the data is moving from one enterprise to another. Another limitation is that the information is stored in clear by the enterprise, and despite the privacy policy the leakage of confidential information can happen due to compromise of the database maintained by the enterprise. Therefore, to solve the aforementioned problem, recent proposals on enforcing access control policies exploit the use of encryption techniques. In such systems, information is stored in an encrypted form by an enterprise and there is no need for an access control manager to check user credentials. Thus, each user can get the encrypted data, but only users who have the right credentials (the right key) can decrypt the encrypted data. This implies that data confidentiality is preserved even if the database maintained by the enterprise is compromised.

Current Solutions (and their Drawbacks) which Enforce Access Control Policies using Encryption

To prevent commercial Web-based PHR systems to access the content of patients PHRs, the patient can encrypt her data using traditional public key encryption algorithms and store the encrypted data (ciphertext) in a database, and then decrypt the ciphertext on demand. In this case, the patient only needs to assume that Microsoft HealthVault and Google Health will properly store her encrypted data, and even if Microsoft HealthVault and Google Health get corrupted, patients PHR will not be disclosed since the data is stored in an encrypted form. The problem with this solution is that the patient needs to be involved in every request (e.g., from her doctor, hospital) and perform the decryption. This is because only the patient knows the decryption key.
One possible solution is to use more advanced public key encryption schemes such as Ciphertext-Policy Attribute-Based Encryption (CP-ABE) scheme [11-13]. The CP-ABE scheme is a type of attribute-based encryption scheme in which the data owner encrypts the data according to an access control policy τ defined over a set of attributes, and where the receiving end can decrypt the encrypted data only if his private key associated with a set of attributes

satisfies the access control policy τ. For example, suppose Alice encrypts her data according to an access policy τ=(a_1 AND a_2) OR a_3. Bob can decrypt the encrypted data only if his private key is associated with a set of attributes that satisfy the access policy. To satisfy the access control policy τ, Bob must have a private key associated with at least one from the following attribute sets: (a_1, a_2), (a_3) or (a_1, a_2, a_3). In CP-ABE the mapping user-attribute is many-to-many, which means that one user may possess many attributes and one attribute may be possessed by many users. The main drawbacks of using CP-ABE to securely manage PHRs are: a) the patient has to know the list of users associated with an attribute, and b) an attribute is possesed by many users. Therefore, using CP-ABE, the patient cannot encrypt a PHR such that only one healthcare provider, with whom the patient has contract, can access it at a time. This is because one attribute can be possesed by many healthcare providers. This is one of the main reasons why both Microsoft HealthVault and Google Health use DAC where access to patients records is based on doctors identity, instead of using Attribute-Based Access Control (ABAC) where access to patients records is based on doctors attributes.
Another solution is to use existing proxy re-encryption schemes [16-20], in which the patient assigns a re-encryption key to the Proxy which re-encrypts the encrypted PHR under patients' public key into encrypted PHR under doctors' public key. However this approach has the following drawbacks:

- The Proxy is able to re-encrypt all ciphertexts of the patient such that the doctor can decrypt all ciphertexts using his private key. Thus, the patient does not have the flexibility to define fine-grained access control.
- If the Proxy and the delegatee get corrupted, then all PHRs may be disclosed to an illegitimate entity based on the fact that the re-encryption key can re-encrypt all ciphertexts.

The Concept of Type-and-Identity-Based Proxy Re-encryption

In order to solve the aforementioned problems a new encryption scheme which would cryptographically enforce access policies is needed. We propose a type-and-identity-based proxy re-encryption scheme which consists of six algorithms : Setup, $\mathrm{Extract}$, $\mathrm{Encrypt}$, $\mathrm{Pextract}$, Preenc and $\mathrm{Decrypt}$.
The basic building block of the type-and-identity-based proxy re-encryption scheme is the

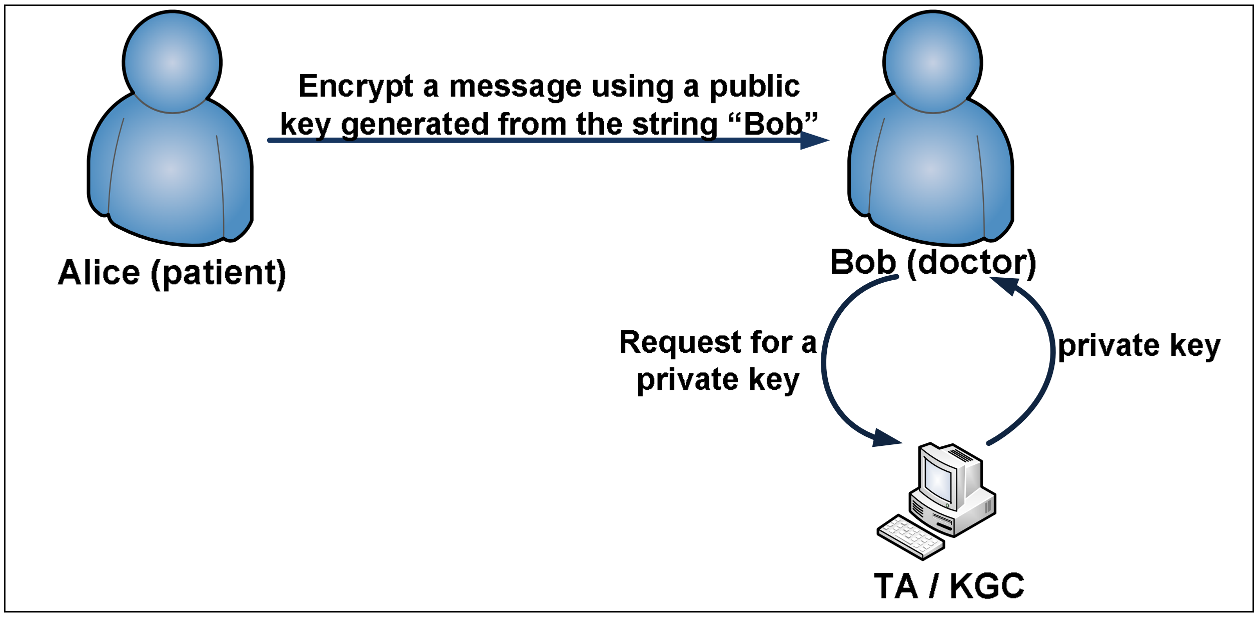
Encrypt a message using a public key generated from the string “Bob”
Alice (patient)
Bob (doctor)
Request for a private key
private key
TA / KGC

Figure 2: Identity-Based Encryption

Identity-Based Encryption (IBE) scheme (Figure 2). The concept of IBE was proposed by Shamir [10] in 1984, however IBE has become practical only after Boneh and Franklin [11] propose the first IBE scheme based on bilinear pairings on elliptic curve (In appendix A we review in more detail the concept of bilinear pairing). The IBE scheme consist of four algorithms: Setup , $\mathrm{Extract}$, $\mathrm{Encrypt}$ and $\mathrm{Decrypt}$. Unlike a traditional public key encryption scheme, an IBE does not require a digital certificate to certify the encryption key (public key) because the public key of any user can be an arbitrary string such as an email address, IP address, etc. Key escrow is an inherent property in IBE systems, i.e., the trusted authority (TA), also referred to as Key Generation Center (KGC), can generate each users' private key, because the TA owns the master key mk used to generate users' private keys. IBE is a very suitable technique to be used in healthcare to exchange emails more securely. For example, in Figure 2, when Alice (the patient) wants to send an encrypted email to Bob (the doctor), Alice can encrypt an email using the encryption key derived from the doctors identity and send the email via an insecure channel. The doctor can authenticate himself to the TA to get the decryption key (private key). After the private key is generated the doctor can decrypt the encrypted email. Unlike in traditional public-key encryption schemes where the private key and the public key has to be created simultaneously, in IBE the private key can be generated long time after the corresponding public key is generated.

A type-and-identity-based proxy re-encryption scheme extends the IBE scheme by adding the Proxy entity to the existing two entities: the TA and users. Another type of extension has been made to the number of algorithms. In addition to the four algorithms of IBE scheme, the type-and-identity-based proxy re-encryption scheme uses $\mathrm{Pextract}$ algorithm to generate a re-encryption key, and Preenc algorithm to re-encrypt the ciphertext. These algorithms are needed to enable the patient (delegator) to specify a fine-grained access control policy for her PHRs.

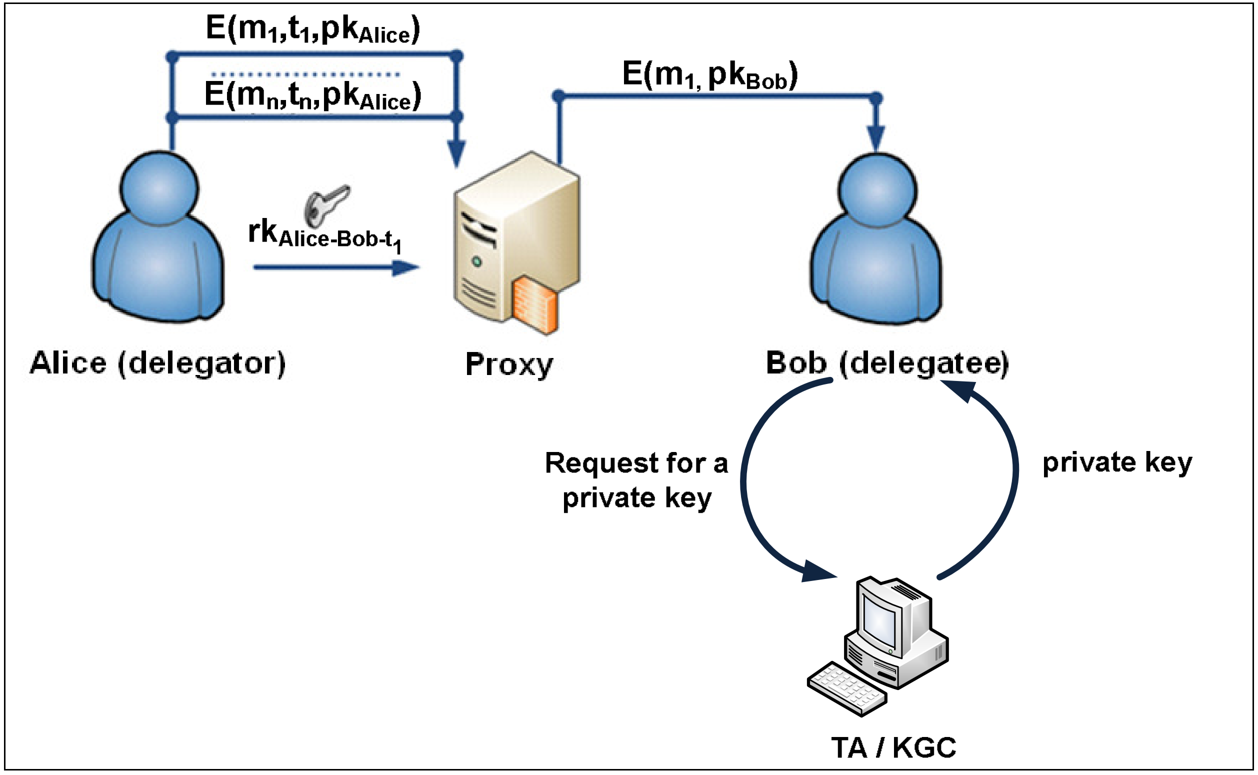

Figure 3: A Type-and-Identity-Based Proxy Re-Encryption

In the type-and-identity-based proxy re-encryption scheme (Figure 3), Alice (delegator) using one key-pair can categorize messages (data) into different types and delegate the decryption right of each type to Bob (delegatee) through a Proxy. Grouping the data into different categories is needed since different PHRs may have different levels of privacy requirements. For example, Alice may not be seriously concerned about disclosing her food statistics to other persons, but she might wish to keep her illness history as a top secret and only disclose it to the appropriate person. In addition to categorizing her PHRs according to the sensitivity level, Alice may categorize her PHRs according to the type of information or according to the device which generated the PHR. There are a number of measurement devices in the market which can be used by the patient and can be connected via home hubs to remote back-end servers. Such examples are disease management services (such as Philips Motiva and PTS) or emergency response services (Philips Lifeline) in which the healthcare provider can remotely access the measurement data and help the patients. As illustrated in Figure 3, Alice uses only one public key to encryp all messages, and delegates her decryption right (computes a re-encryption key) only for one type (type 1), and Bob can use his private key to access only messages which belong to type 1. If the Proxy and Bob get corrupted, then only health records belonging to type 1 may be disclosed to an illegitimate entity, while the other types of information remains secure. A full construction of the type-and-identity based re-encryption scheme is given in Appendix B.

The six algorithms are defined as follows:

- $\mathrm{Setup}(k)$: run by the TA, the algorithm takes as input a security parameter k and outputs the master public key pk and the master private key mk. pk is used in the encryption phase by each user, and mk is used by the TA to generate users private keys.

- $\text{Extract}(mk, id)$: run by the TA when a user request a private key. The algorithm takes as input the master key mk and an user identity id, the algorithm outputs the user private key sk_{id}.
- $\text{Encrypt}(m, t, pk_{id})$: run by the encryptor, the algorithm takes as input a message to be encrypted m, a type t, and an the public key pk_{id} associated with the identity id, and outputs the ciphertext c.
- $\text{Pextract}(id_i, id_j, t, sk_{id_i})$ run by the delegator, this algorithm takes the delegator's identifier id_i, the delegatee's identifier id_j, the type t, and the delegator's private key sk_{id_i} as input and outputs the re-encryption key $rk_{id_i \to id_j}$.
- $\text{Preenc}(c_i, rk_{id_i \to id_j})$ run by the Proxy, this algorithm, takes as input the ciphertext c_i and the re-encryption key $rk_{id_i \to id_j}$, and outputs a new ciphertext c_j
- $\text{Decrypt}(c_j, sk_{id_j})$ run by the decryptor, the algorithm takes as input the ciphertext c_j and the private key sk_{id_j}, and output a message m.

A formal security model and a formal security proof is given in appendix B.1. and B.2.

Applying the Scheme in Practice

Figure 4 illustrates a general architecture of a PHR system that uses our type-and-identity-based proxy re-encryption scheme. The architecture consists of Trusted Authorities (TAs), a patient, an application hosting device, a Web PHR, a Proxy and a doctor. The TAs are used to generate key pairs for the patient, respectively the doctor. We assume that the patient and the doctor are from different security domains. The application hosting device can be implemented on a home PC of the data source (a patient) or as a trusted service. Its role is to encrypt PHRs and forward them to the Web PHR. The Web PHR stores encrypted PHRs, and Proxy is used to re-encrypt encrypted data and forward them to the doctor. There are five basic processes in the management of PHRs:

- Setup: In this phase, TAs run the Setup and Extract algorithm and distribute the public parameters needed to run the algorithms of the type-and-identity-based proxy re-encryption scheme, and distributes the private keys, which are needed to decrypt encrypted messages, to the patient and doctor (1). We assume that there is a secure channel between the TA and the user, respectively the doctor. Note that the doctor can get his key pair during the decryption phase, while the Proxy can perform re-encryption of encrypted data under doctors public key, even if the doctor does not have a private key. This is possible since the doctors public key can be computed by everyone who knows doctors' identity. The TA does not have to be online as long as each user gets his/her key pair.
- Data creation: The patient uses a number of healthcare devices and creates measurement data and forwards them to the application hosting device (2). In addition

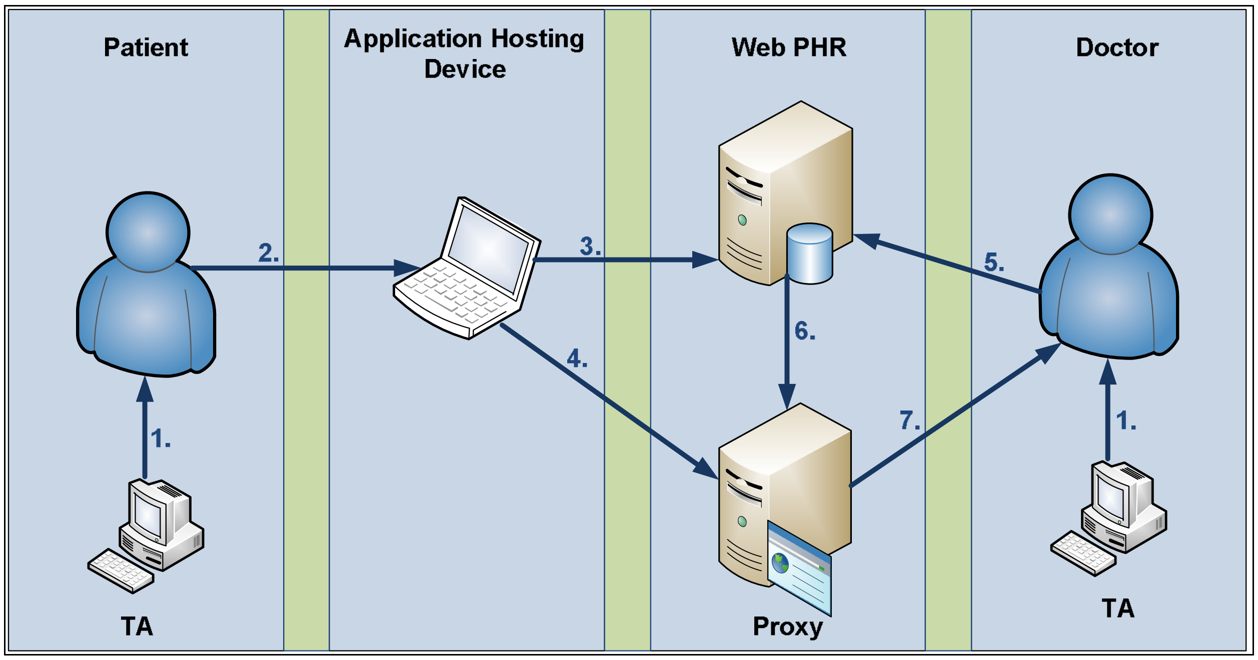

Figure 4: Secure Management of PHR

to that, the patient can forward to the application hosting device all kinds of information that the patient has from various medical service providers.

- Data protection: The patient categorizes her PHR according to her privacy concerns. For instance, she can set her illness history as type t_1, her food statistics as type t_2, and the necessary PHR data in case of emergency as type t_3. Then the patient generates the encryption key (public key) derived from her identity (identity can be any type of public string) and run the $\mathrm{Encrypt}$ algorithm using the generated public key. After the encryption is performed, the patients uploads the encrypted data to Web PHR (3). As in the previous step, all this (data categorization, data encryption and data uploading) can be done by a hosting device on behalf of the patient.
- Data Sharing (allow the doctor to see patient data): In this phase, the patient runs the $\mathrm{Pextract}$ algorithm and generates the re-encryption key which will be used by the Proxy to re-encrypt encrypted data under patients public key to encrypted data under doctors public key, such that the doctor can decrypt the encrypted data using his private key. The generated key is forwarded to the Proxy (4). As in the above steps, all this can be done by a hosting device on behalf of the patient who specifies the access control policy.
- Data consumption (doctors' request-response): When a doctor wants to use patient data, he contacts the Web PHR and specifies the ciphertext that he wants to decrypt (5). We assume that each ciphertext is associated with appropriate metadata - descriptive information about the patient. The encrypted data is forwarded to the Proxy (6). The Proxy checks if the doctor is allowed to see patient data (checks if it has a re-encryption key $rk_{Patient \to Doctor}$), and, if so, the Proxy runs the Preenc algorithm to re-encrypt the

encrypted data. The re-encrypted data is sent to the doctor (7). After receiving the re-encrypted data the doctor runs the $\mathrm{Decrypt}$ algorithm using his private key.

Trust Assumptions

User trust is very important aspect when deploying a Web-based PHR system. In practice, users have greater trust on systems where they can control access to their information, and lower trust on systems where they have to trust someone else to control access to their information (e.g., the user has to trust the access control manager of the Web PHR). Therefore in this paper we provide a solution which compared to existing solutions reduces the trust that patients need to have on commercial Web-based PHR systems. As mentioned above, in our proposal the role of the Web PHR is twofold: a) to provide storage for PHRs, and b) to maintain the Proxy. Next to that, the patient has to put the following trust on Web PHR:

- The Web-based PHR system is trusted to store PHRs in publicly accessible storage only in an encrypted form, therefore the patient does not have to trust the Web-based PHR system to provide data confidentiality service. The data confidentiality service is provided by the patient at the moment when the data is encrypted. Encryption prevents sniffing software to access the data when the data travels from the user to the storage, and the storage cannot decrypt the data without having the private key.
- The Proxy is trusted to maintain a list of re-encryption keys, and to enforce the access policy by properly re-encrypting the encrypted data when the identity of the doctor (requester) is part of the re-encryption key. Note that, the list of re-encryption keys should be secret (if the list of re-encryption keys is public then the patient cannot prevent an authorized doctor to see her data after the access decision is made) , therefore the patient has to trust the Proxy to store all re-encryption keys securely. The difference between our approach and the access control mechanisms in existing Web PHR is that in our approach the Proxy who plays the role of the ACM cannot access the content of PHRs, therefore, the patient does not have to fully trust the Proxy, while in existing commercial Web-based PHR systems the patient has to fully trust the ACM because the ACM can access the content of PHRs.
- The user should trust the Proxy to securely delete re-encryption keys when the user wants to prevent an authorized doctor to access users data further. For example, a patient might change her healthcare provider, and after some time she wants to prevent the doctor from an old healthcare provider to access her data.

Policy Updating

For patient to allow someone to access her data, the patient has to compute a new re-encryption key. For example, if Alice wants to allow Bob and Charlie to see her PHRs belonging to category t_{Alergy}, Alice has to create two re-encryption keys: $rk_{Alice \to Bob}$ and $rk_{Alice \to Charlie}$. Transmission of the re-encryption keys to the Proxy should be secured using cryptographical protocols such as Transport Layer Security (TLS) which allows two entities to securely

communicate over the internet.
In practice the patient might want to update her access policy. Using our approach the patient might do this task without entirely decrypting the ciphertext. To update an access policy means to create new, or delete old, re-encryption keys. For example, Alice to update her access policy from $\tau = Bob\ OR\ Charlie$ (this access policy implies that Bob and Charlie can access the data) to a different access policy $\tau' = Bob\ OR\ Dave$ (this access policy implies that Bob and Dave can access the data), has to follow the following two procedures:

- Notify the Proxy to delete (revoke) the re-encryption key $rk_{Alice \to Bob}$. This would prevent the Proxy to re-encrypt encrypted data under Alices public key to encrypted data under Bobs public key.
- Create new re-encryption key $rk_{Alice \to Charlie}$ and send it to the Proxy. This would allow the Proxy to re-encrypt encrypted data under Alices public key to encrypted data under Daves public key.

<u>Related Work on Proxy Re-Encryption</u>

Since Mambo and Okamoto first proposed the concept [12], a number of proxy re-encryption schemes have been proposed [16-20]. Blaze *et al.* [17] introduce the concept of atomic Proxy cryptography which is the current concept of proxy re-encryption. In a proxy re-encryption scheme, the Proxy can transform ciphertexts encrypted with the delegator's public key into ciphertexts that can be decrypted with the delegatee's private key. Blaze *et al.* propose a proxy re-encryption scheme based on the ElGamal encryption scheme [18]. One property of this scheme is that, with the same re-encryption key, the Proxy can transform the ciphertexts not only form the delegator to the delegatee but also from the delegatee to the delegator. This is called the “bi-directional” property in the literature. Bi-directionality might be a problem in some applications, but it might also be a desirable property in some other applications. Jacobsson [14] addresses this “problem” using a quorum controlled asymmetric proxy re-encryption where the Proxy is implemented with multiple servers and each of them performs partial re-encryption.
Dodis and Ivan [13] propose a generic construction method for proxy re-encryption schemes and also provide a number of example schemes. Their constructions are based on the concept of secret splitting, which means that the delegator splits his private key into two parts and sends them to the Proxy and the delegatee separately. During the re-encryption process the Proxy performs partial decryption of the encrypted message using the first part of the delegator's private key, and the delegatee can recover the message by performing partial decryption using the second part of the delegator's private key. One disadvantage of this method is that it is not collusion-safe, i.e. the Proxy and the delegatee together can recover the delegator's private key. Another disadvantage of this scheme is that the delegatee's public/private key pair can only be used for dealing with the delegator's messages. If this key pair is used by the delegatee for other encryption services, then the delegator can always decrypt the ciphertexts.
Ateniese *et al.* [19] propose several proxy re-encryption schemes based on the ElGamal

scheme. In their schemes, the delegator does not have to interact and share his private key with the delegatee. The delegator stores two private keys, a master private key and a "weak" private key. The ciphertext can be fully decrypted using either of the two distinct keys. Their scheme is collusion safe, since only the "weak" private key is exposed if the delegatee and the Proxy collude but the master key remains safe. The disadvantage of this scheme is that the delegator has to perform two levels of encryptions, the first level encryption encrypts messages that can be decrypted by the delegator, and the second level encryption encrypts messages that can be decrypted by the delegator and his delegatees. In addition, Ateniese *et al.* also discuss a number of properties for proxy re-encryption schemes in [19].
Recently, two IBE proxy re-encryption schemes were proposed by Matsuo [16] and Green and Atteniese [15], respectively. The Matsuo scheme assumes that the delegator and the delegatee belong to the same KGC and use the Boneh-Boyen encryption scheme [20]. The Green-Atteniese scheme assumes that the delegator and the delegatee can belong to different KGCs but the delegatee possess the public parameter of the delegator's KGC.
Sahai and Waters [21] introduce the concept of Attribute-Based Encryption (ABE) which is a generalized form of IBE. In ABE the ciphertext and user private keys are associated with a set of attributes. A user can decrypt the ciphertext if the user private key has the list of attributes specified in the ciphertext. In Ciphertext-Policy Attribute-Based Encryption (CP-ABE) [11-13] the user private key is associated with a set of attributes and a ciphertext is associated with an access control over some attribute. The decryptor can decrypt the ciphertext if the list of attributes associated with the private key satisfies the access policy. In Key-Policy Attribute-Based Encryption (KP-ABE) [22] the idea is reversed and the private key is associated with an access control over some attributes and the ciphertext is associated with a list of attributes. The decryptor can decrypt the ciphertext if the list of attributes associated with the ciphertext satisfy the access policy associated with the private key. Lliang et al.[23] proposed an attribute-based proxy re-encryption scheme. The scheme is based on the Cheung and Newport CP-ABE scheme [8] and inherits the same limitations that [8] has: it supports only access policies with AND boolean operator, and the size of the ciphertext increases linearly with the number of attributes in the system.
Proxy re-encryption has many promising applications including access control in file storage [19], email forwarding [24], and law enforcement [12]. With the increasing privacy concerns over personal data, proxy re-encryption, in particular IBE proxy re-encryption schemes (due to their benefits [11]), will find more and more applications. As we show in this paper, proxy re-encryption is a powerful tool for patients to enforce their PHR disclosure policies.

Conclusion

This paper presents a new approach for secure management of PHRs which are stored and shared from a semitrusted web server (the server is trusted to perform only the ciphertext re-encryption, without having access to the plain text). We gave an overview of access control mechanisms employed in current commercial Web-based PHR systems and show that traditional access control mechanisms as well as traditional encryption techniques which enforce access control policies are not suitable to be used in scenarios where the data is outsourced to a third party data center. In this paper we propose a type-and-identity-based proxy re-encryption scheme which allow patients to reduce the trust on commercial Web-base PHR systems and enable patients to provide different re-encryption capabilities to the Proxy while using the same key pair. This property is showed to be useful in our PHR disclosure system, where an individual can easily implement fine-grained access control policies to her PHRs.

References:

[1] The US Department of Health and Human Services. Summary of the HIPAA privacy rule. http://www.hhs.gov/ocr/privacy/hipaa/understanding/summary/privacysummary.pdf, 2003. Accessed on 2 October, 2009.

[2] Microsoft HealthVault. http://www.healthvault.com/. Accessed on 15 September, 2009.

[3] Google Health. http://www.google.com/health. Accessed on 15 September, 2009.

[4] The personal health working group final report. Connecting for health. http://www.connectingforhealth.org/resources/wg_eis_final_report_0704.pdf, 2004. Accessed on 2 October, 2009.

[5] 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.

[6] G.S. Graham and P.J. Denning. Protection: principles and practice. In *Proceedings of the November 16-18, 1971, fall joint computer conference*, pages 417–429. ACM New York, NY, USA, 1971.

[7] J. Bethencourt, A. Sahai, and B. Waters. Ciphertext-policy attribute-based encryption. pages 321–334. IEEE Computer Society Washington, DC, USA, 2007.

[8] L. Cheung and C. Newport. Provably secure ciphertext policy ABE. pages 456–465. ACM New York, NY, USA, 2007.

[9] L. Ibraimi, Q. Tang, P. Hartel, and W. Jonker. Efficient and provable secure ciphertext-policy attribute-based encryption schemes. In *Information Security Practice and Experience*, volume 5451, pages 1–12. Springer, 2009.

[10] A. Shamir. Identity-based cryptosystems and signature schemes. *Proceedings of CRYPTO 84 on Advances in cryptology table of contents*, pages 47–53, 1985.

[11] 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.

[12] 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.

[13] A. Ivan and Y. Dodis. Proxy cryptography revisited. In *Proceedings of the Network and Distributed System Security Symposium*. The Internet Society, 2003.

[14] Markus Jakobsson. On quorum controlled asymmetric proxy re-encryption. In H. Imai and Y. Zheng, editors, *Public Key Cryptography, Second International Workshop on Practice and*

Theory in Public Key Cryptography, volume 1560 of *Lecture Notes in Computer Science*, pages 112–121. Springer, 1999.

[15] 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.

[16] T. Matsuo. Proxy re-encryption systems for identity-based encryption. In T. Takagi, T. Okamoto, E. Okamoto, and T. Okamoto, editors, *Pairing-Based Cryptography - Pairing 2007, First International Conference*, volume 4575 of *Lecture Notes in Computer Science*, pages 247–267. Springer, 2007.

[17] 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.

[18] T. ElGamal. A public key cryptosystem and a signature scheme based on discrete logarithms. In G. R. Blakley and D. Chaum, editors, *Advances in Cryptology, Proceedings of CRYPTO '84*, volume 196 of *Lecture Notes in Computer Science*, pages 10–18. Springer, 1985.

[19] 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 (TISSEC)*, 9(1):1–30, 2006.

[20] D. Boneh and X. Boyen. Efficient selective-id secure identity-based encryption without random oracles. In C. Cachin and J. Camenisch, editors, *Advances in Cryptology - EUROCRYPT 2004, International Conference on the Theory and Applications of Cryptographic Techniques*, volume 3027 of *Lecture Notes in Computer Science*, pages 223–238. Springer, 2004.

[21] A. Sahai and B. Waters. Fuzzy identity-based encryption. In *Advances in Cryptology–Eurocrypt 2005*, volume 3494, pages 457–473. Springer, 2005.

[22] V. Goyal, O. Pandey, A. Sahai, and B. Waters. Attribute-based encryption for fine-grained access control of encrypted data. pages 89–98. ACM Press New York, NY, USA, 2006.

[23] X. Liang, Z. Cao, H. Lin, and J. Shao. Attribute based proxy re-encryption with delegating capabilities. In *Proceedings of the 4th International Symposium on Information, Computer, and Communications Security*, pages 276–286. ACM New York, NY, USA, 2009.

[24] 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.

[25] L. Chen. An interpretation of identity-based cryptography. In A. Aldini and R. Gorrieri, editors, *Foundations of Security Analysis and Design IV, FOSAD 2006/2007 Tutorial Lectures*, volume 4677 of *Lecture Notes in Computer Science*, pages 183–208. Springer, 2007.

[26] V. Shoup. Sequences of games: a tool for taming complexity in security proofs. http://shoup.net/papers/, 2006.

Appendix

A. Review of pairing

We briefly review the basis of pairing and the related assumptions. More detailed information can be found in the seminal paper [11]. A pairing (or, bilinear map) satisfies the following properties:

1. $\mathbb{G}$ and $\mathbb{G}_1$ are two multiplicative groups of prime order p;
2. g is a generator of $\mathbb{G}$;
3. $\hat{e}:\mathbb{G}\times\mathbb{G}\rightarrow\mathbb{G}_1$ is an efficiently-computable bilinear map with the following properties:
   - Bilinear: for all $u,v\in\mathbb{G}$ and $a,b\in\mathbb{Z}_p^*$, we have $\hat{e}(u^a,v^b)=\hat{e}(u,v)^{ab}$.
   - Non-degenerate: $\hat{e}(g,g)\neq 1$.

As defined in [11], $\mathbb{G}$ is said to be a bilinear group if the group action in $\mathbb{G}$ can be computed efficiently and if there exists a group $\mathbb{G}_1$ and an efficiently-computable bilinear map $\hat{e}$ as defined above.

The Bilinear Diffie-Hellman (BDH) problem in $\mathbb{G}$ is as follows: given $g,g^a,g^b,g^c\in\mathbb{G}$ as input, output $\hat{e}(g,g)^{abc}\in\mathbb{G}_1$. An algorithm A has advantage ε in solving BDH in $\mathbb{G}$ if:

$$\Pr[\mathsf{A}(g,g^a,g^b,g^c)=\hat{e}(g,g)^{abc}]\geq\varepsilon.$$

Similarly, we say that an algorithm A has advantage ε in solving the decision BDH problem in $\mathbb{G}$ if:

$$|\Pr[\mathsf{A}(g,g^a,g^b,g^c,g^{abc})=0]-\Pr[\mathsf{A}(g,g^a,g^b,g^c,T)=0]|\geq\varepsilon.$$

Here the probability is over the random choice of $a,b,c\in\mathbb{Z}_p^*$, the random choice of $T\in\mathbb{G}_1$, and the random bits of A (the adversary is a nondeterministic algorithm).

Definition 1. *We say that the (decision) (t,ε)-BDH assumption holds in $\mathbb{G}$ if no t-time algorithm has advantage at least ε in solving the (decision) BDH problem in $\mathbb{G}$.*

As in the general group, the Computational Diffie-Hellman (CDH) problem in $\mathbb{G}$ is as follows: given $g,g^a,g^b\in\mathbb{G}$ as input, output $g^{ab}\in\mathbb{G}$. An algorithm A has advantage ε in solving CDH in $\mathbb{G}$ if:

$$\Pr[\mathsf{A}(g,g^a,g^b)=g^{ab}]\geq\varepsilon.$$

Definition 2. *We say that the (t,ε)-CDH assumption holds in $\mathbb{G}$ if no t-time algorithm has advantage at least ε in solving the CDH problem in $\mathbb{G}$.*

Given a security parameter k, a problem (say, BDH) is believed to be intractable if any adversary has only negligible advantage in reasonable time. We usually define a scheme to be secure if any adversary has only a negligible advantage in the underlying security model. The time parameter is usually be ignored.

Definition 3. *The function $P(k):\mathbb{Z}\rightarrow\mathbb{R}$ is said to be negligible if, for every polynomial $f(k)$, there exists an integer N_f such that $P(k)\leq\frac{1}{f(k)}$ for all $k\geq N_f$.*

B. Our Construction

In this section we give the construction of the type-and-identity-based proxy re-encryption scheme. In our scheme, the delegator and the delegatee are allowed to be from different domains, which nonetheless share some public parameters.

- Suppose that the delegator is registered at KGC$_1$ in Boneh-Franklin IBE scheme $(\mathrm{Setup}_1, \mathrm{Extract}_1, \mathrm{Encrypt}_1, \mathrm{Decrypt}_1)$ and the delegatee is registered at KGC$_2$ in Boneh-Franklin IBE scheme $(\mathrm{Setup}_2, \mathrm{Extract}_2, \mathrm{Encrypt}_2, \mathrm{Decrypt}_2)$. The algorithms are defined as follows.

- Setup_1 and $\mathrm{Extract}_1$ are the same as in the Boneh-Franklin scheme, except that Setup_1 outputs an additional hash function $\mathrm{H}_2 : \{0,1\}^* \to \mathbb{Z}_p^*$. The public parameter is $params_1 = (\mathbb{G}, \mathbb{G}_1, p, g, \mathrm{H}_1, \mathrm{H}_2, \hat{e}, pk_1)$, and the master key is $mk_1 = \alpha_1$.
- $\mathrm{Encrypt}_1(m,t,id)$: Given a message m, a type t, and an identifier id, the algorithm outputs the ciphertext $c = (c_1, c_2, c_3)$ where $r \in_R \mathbb{Z}_p^*$,

$$c_1 = g^r,\ c_2 = m \cdot \hat{e}(pk_{id}, pk)^{r \cdot \mathrm{H}_2(sk_{id} \| t)},\ c_3 = t.$$

- $\mathrm{Decrypt}_1(c, sk_{id})$: Given a ciphertext $c = (c_1, c_2, c_3)$, the algorithm outputs the message

$$m = \frac{c_2}{\hat{e}(sk_{id}, c_1)^{\mathrm{H}_2(sk_{id} \| c_3)}}$$

Without loss of generality, suppose the delegator holds the identity id_i and the corresponding private key sk_{id_i}. Apart from the delegator, another party cannot run the $\mathrm{Encrypt}_1$ algorithm under the delegator's identity id_i since he does not know sk_{id_i}.

-Suppose that the delegatee (with identity id_j) possesses private key sk_{id_j} registered at KGC$_2$ in the Boneh-Franklin IBE scheme, where the public parameter is $params_2 = (\mathbb{G}, \mathbb{G}_1, p, g, \mathrm{H}_1, \hat{e}, pk_2)$, the master key is $mk_2 = \alpha_2$, and $sk_{id_j} = \mathrm{H}_1(id_j)^{\alpha_2}$. For the ease of comparison, we denote the IBE scheme as $(\mathrm{Setup}_2, \mathrm{Extract}_2, \mathrm{Encrypt}_2, \mathrm{Decrypt}_2)$ although these algorithms are identical to those described in Section B.

- The delegation process

If the delegator wants to delegate his decryption right for messages with type t to the delegatee, the algorithms of the proxy re-encryption scheme are as follows.

- $\mathrm{Pextract}(id_i, id_j, t, sk_{id_i})$: Run by the delegator, this algorithm outputs the re-encryption key $rk_{id_i \to id_j}$, where $X \in_R \mathbb{G}_1$ and

$$rk_{id_i \to id_j} = (t, sk_{id_i}^{-\mathrm{H}_2(sk_{id_i} \| t)} \cdot \mathrm{H}_1(X), \mathrm{Encrypt}_2(X, id_j)).$$

- $\mathrm{Preenc}(c_i, rk_{id_i \to id_j})$: Run by the Proxy, this algorithm, takes a ciphertext $c_i = (c_{i1}, c_{i2}, c_{i3})$ and the re-encryption key $rk_{id_i \to id_j}$ as input where $t = c_{i3}$, and outputs a new ciphertext

$c_j = (c_{j1}, c_{j2}, c_{j3})$, where $c_{j1} = c_{i1}$ and

$$\begin{aligned} c_{j2} &= c_{i2} \cdot \hat{e}(c_{i1}, sk_{id_i}^{-\mathrm{H}_2(sk_{id_i} \| c_{i3})} \cdot \mathrm{H}_1(X)) \\ &= m \cdot \hat{e}(g^{\alpha_1}, pk_{id_i}^{r\mathrm{H}_2(sk_{id_i} \| t)}) \cdot \hat{e}(g^r, sk_{id_i}^{-\mathrm{H}_2(sk_{id_i} \| t)} \cdot \mathrm{H}_1(X)) \\ &= m \cdot \hat{e}(g^r, \mathrm{H}_1(X)), \end{aligned}$$

and $c_{j3} = \mathrm{Encrypt}_2(X, id_j)$.

- $\mathrm{Decrypt}(c_j, sk_{id_j})$.Given a re-encrypted ciphertext c_j, the delegatee can obtain the plaintext m by computing:

$$\begin{aligned} m' &= \frac{c_{j2}}{\hat{e}(c_{j1}, \mathrm{H}_1(\mathrm{Decrypt}_2(c_{j3}, sk_{id_j})))} \\ &= \frac{m \cdot \hat{e}(g^r, \mathrm{H}_1(X))}{\hat{e}(g^r, \mathrm{H}_1(X))} \\ &= m. \end{aligned}$$

B.1. Security Model

We assume that the Proxy is semi-trusted in the following sense: it will honestly convert the delegator's ciphertexts using the re-encryption key; however, it might act actively to obtain some information about the plaintexts for the delegator and the delegatee. As mentioned in [25], the key escrow problem (TA owns a master key which can be used to decrypt each encrypted data) can be avoided by applying some standard techniques (such as secret sharing) to the underlying scheme, hence, we skip any further discussion in this paper. The delegatee may be curious in the sense that it may try to obtain some information about the plaintexts corresponding to the delegator's ciphertexts which have not been re-encrypted by the Proxy.
As a standard practice, we describe an attack game for modeling the semantic security against an adaptive chosen plaintext attack for the delegator (IND-ID-DR-CPA security) for our scheme. The IND-ID-DR-CPA game is carried out between a challenger and an adversary, where the challenger simulates the protocol execution and answers the queries from the adversary. Note that the allowed queries for the adversary reflect the adversary's capability in practice. Specifically, the game is as follows:
1. Game setup: The challenger takes a security parameter k as input, runs the Setup_1 algorithm to generate the public system parameter $params_1$ and the master key mk_1, and runs the Setup_2 algorithm to generate the public system parameter $params_2$ and the master key mk_2.
2. Phase 1: The adversary takes $params_1$ and $params_2$ as input and is allowed to issue the following types of queries:
   a) $\mathrm{Extract}_1$ query with any identifier id: The challenger returns the private key sk corresponding to id.
   b) $\mathrm{Extract}_2$ query with any identifier id': The challenger returns the private key sk' corresponding to id'.
   c) $\mathrm{Pextract}$ query with (id, id', t): The challenger returns the re-encryption key $rk_{id \to id'}$ for the type t.

d) $\mathrm{Preenc}^{\dagger}$ query with (m,t,id,id'): The challenger first computes $c=\mathrm{Encrypt}_1(m,t,id)$ and then returns a new ciphertext c' which is obtained by applying the delegation key $rk_{id\to id'}$ to c, where $rk_{id\to id'}$ is issued for type t.

Once the adversary decides that Phase 1 is over, it outputs two equal length plaintexts m_0,m_1, a type t^*, and an identifier id^*. At the end of Phase 1, there are three constraints here:

a) id^* has not been the input to any $\mathrm{Extract}_1$ query.

b) For any id', if (id^*,id',t^*) has been the input to a $\mathrm{Pextract}$ query then id' has not been the input to any $\mathrm{Extract}_2$ query.

c) If there is a $\mathrm{Preenc}^{\dagger}$ query with (m,t,id,id'), then (id,id',t) has not been queried to $\mathrm{Pextract}$.

3. Challenge: The challenger picks a random bit $b\in\{0,1\}$ and returns $c^*=\mathrm{Encrypt}_1(m_b,t^*,id^*)$ as the challenge to the adversary.
4. Phase 2: The adversary is allowed to continue issuing the same types of queries as in Phase 1. At the end of Phase 2, there are the same constraints as at the end of Phase 1.
5. Guess (game ending): the adversary outputs a guess $b'\in\{0,1\}$.

At the end of the game, the adversary's advantage is defined to be $|\Pr[b'=b]-\frac{1}{2}|$. Compared with the CPA security formalizations in [19,20], in our case, we also take into account the categorization of messages for the delegator. The $\mathrm{Preenc}^{\dagger}$ query reflects the fact that a curious delegatee has access to the the delegator's plaintexts.

B.2. Security Proof

Theorem 1. *For the type-and-identity-based proxy re-encryption scheme described in Section B, any adversary's advantage is negligible.*

Proof sketch. We suppose that the total number of queries issued to H_1 and H_2 is bounded by integer q_1 and q_2, respectively. Suppose an adversary A has the non-negligible advantage ε in the IND-ID-DR-CPA game. The security proof is done through a sequence of games.

Game_0: In this game, B faithfully answers the oracle queries from A. Specifically, B simulates the random oracle H_1 as follows: B maintains a list of vectors, each of them containing a request message, an element of G (the hash-code for this message), and an element of Z_p^*. After receiving a request message, B first checks its list to see whether the request message is already in the list. If the check succeeds, B returns the stored element of G; otherwise, B returns g^y, where y a randomly chosen element of Z_p^*, and stores the new vector in the list. A' simulates the random oracle H_2 as follows: B maintains a list of vectors, each of them containing a request message and an element of Z_p^* (the hash-code for this message). After receiving a request message, B first checks its list to see whether the request message is already in the list. If the check succeeds, B returns the stored element of Z_p^*; otherwise, B returns u which is a randomly chosen element of Z_p^*, and stores the new vector in the list.

Let $\delta_0=\Pr[b'=b]$, as we assumed at the beginning, $|\delta_0-\frac{1}{2}|=\varepsilon$.

Game_1: In this game, B answers the oracle queries from A as follows.

- Game setup: B faithfully simulates the setup phase.
- Phase 1: B randomly selects $j \in \{1,2,\cdots,q_1+1\}$. If $j = q_1+1$, B faithfully answers the oracle queries from A. If $1 \le j \le q_1$, we assume the j-th input to H_1 is $\tilde{i}d$ and B answers the oracle queries from A as follows: Answer the queries to $\mathrm{Extract}_1$, $\mathrm{Extract}_2$, $\mathrm{Pextract}$, and $\mathrm{Preenc}^{\dagger}$ faithfully, except that B aborts as a failure when $\tilde{i}d$ is the input to a $\mathrm{Extract}_1$ query.
- Challenge: After receiving (m_0, m_1, t^*, id^*) from the adversary, if one of the following events occurs, B aborts as a failure.
  - (a) id^* has been issued to H_1 as the i-th query and $i \ne j$,
  - (b) id^* has not been issued to H_1 and $1 \le j \le q_1$.

Note that, if the adversary does not abort then either $1 \le j \le q_1$ and $id^* = \tilde{i}d$ is the input to j-th H_1 query or $j = q_1+1$ and id^* has not been the input to any H_1 query. B faithfully returns the challenge.

- Phase 2: B answers the oracle queries faithfully.
- Guess (game ending): the adversary outputs a guess $b' \in \{0,1\}$.

The probability that B successfully ends is $\frac{1}{q_1+1}$, i.e. the probability that B does not abort in its execution is $\frac{1}{q_1+1}$. Let $\delta_1 = \Pr[b' = b]$ when B successfully ends, in which case $|\delta_1 = \delta_0|$. Let θ_1 be the probability that B successfully ends and $b' = b$. We have $\theta_1 = \frac{\delta_1}{q_1+1}$.

Game_2: In this game, B simulates the protocol execution and answers the oracle queries from A in the following way.

- Game setup: B faithfully simulates the setup phase. Recall that $pk_1 = g^{\alpha_1}$.
- Phase 1: B randomly selects $j \in \{1,2,\cdots,q_1+1\}$. If $j = q_1+1$, B faithfully answers the oracle queries from A. If $1 \le j \le q_1$, B answers j-th query to H_1 with g^{β} where $\beta \in_R \mathbb{Z}_p^*$, and answers the oracle queries from A as follows. Suppose the input of the j-th query to H_1 is $\tilde{i}d$.

a) Answer $\mathrm{Extract}_1$ and $\mathrm{Extract}_2$ faithfully, except that B aborts as a failure when $\tilde{i}d$ is the input to a $\mathrm{Extract}_1$ query.

b) $\mathrm{Pextract}$ query with (id, id', t): If $id = \tilde{i}d$, B returns the re-encryption key $rk_{id \to id'}$, where

$$g_{t:id'} \in_R \mathbb{G},\ X_{t:id'} \in_R \mathbb{G}_1,\ rk_{id \to id'} = (t, g_{t:id'}, \mathrm{Encrypt}_2(X_{t:id'}, id')).$$

Otherwise, B answers the query faithfully. If id' has been queried to $\mathrm{Extract}_2$, when $X_{t:id'}$ is queried to H_1 then B returns $g_{t:id'} \cdot h_{t:id'}^{-1}$ where $h_{t:id'} \in_R \mathbb{G}$.

c) $\mathrm{Preenc}^{\dagger}$ query with (m, t, id, id'): If $id = \tilde{i}d$, B returns

$$r \in_R \mathbb{Z}_p^*,\ X_{t:id'} \in_R \mathbb{G}_1,\ c' = (g^r, \hat{e}(g^r, \mathrm{H}_1(X_{t:id'})), \mathrm{Encrypt}_2(X_{t:id'}, id')).$$

Otherwise, B answers the query faithfully.

- Challenge: After receiving (m_0, m_1, t^*, id^*) from the adversary, if one of the following events occurs, B aborts as a failure.
  - (a) id^* has been issued to H_1 as the i-th query and $i \ne j$,
  - (b) id^* has not been issued to H_1 and $1 \le j \le q_1$.

Note that, if the adversary does not abort then either $1 \le j \le q_1$ and $id^* = \tilde{i}d$ is the input to j-th H_1 query or $j = q_1 + 1$ and id^* has not been the input to any H_1 query. In the latter case, B sets $\mathrm{H}_1(id^*) = g^{\beta}$ where $\beta \in_R \mathbb{Z}_p^*$, and returns $c^* = (c_1^*, c_2^*, c_3^*)$ as the challenge to the adversary, where:

$$b \in_R \{0,1\}, r \in_R \mathbb{Z}_p^*, T \in_R \mathbb{G}_1, c_1^* = g^r, c_2^* = m_b \cdot T, c_3^* = t^*.$$

- Phase 2: B answers the oracle queries from A as in Phase 1.
- Guess (game ending): the adversary outputs a guess $b' \in \{0,1\}$.

Let θ_2 be the probability that B successfully ends and $b' = b$. We have $\theta_2 = \frac{1}{2(q_1+1)}$ since $T \in_R \mathbb{G}_1$. Let E_1 be the event that, for some id' and t, the adversary issues a H_2 query with the input $g^{\alpha_1 \beta} \| t$ or $X_{t\|id'}$ is issued to H_1 while id' has not been issued to $\mathrm{Extract}_2$. Compared with Game_1, Game_2 differs when E_1 occurs. From the difference lemma [26], we have $|\delta_2 - \delta_1| \le \varepsilon_2$ which is negligible in the random oracle model based on the BDH assumption. Note that $(\mathrm{Setup}_2, \mathrm{Extract}_2, \mathrm{Encrypt}_2, \mathrm{Decrypt}_2)$ is one-way based on the BDH assumption and BDH implies CDH. From $|\theta_2 - \theta_1| \le \varepsilon_2$ and $\theta_2 = \frac{1}{2(q_1+1)}$, we have $|\frac{1}{2(q_1+1)} - \theta_1| \le \varepsilon_2$. In addition, from $|\delta_0 - \frac{1}{2}| = \varepsilon$, $|\delta_1 - \delta_0| \le \varepsilon_1$ and $\theta_1 = \frac{\delta_1}{q_1+1}$, we have $\frac{\varepsilon}{q_1+1} \le \frac{\varepsilon_1}{q_1+1} + \varepsilon_2$. Because ε_i $(1 \le i \le 2)$ are negligible and ε is assumed to be non-negligible, we get a contradiction. As a result, the proposed scheme is IND-ID-DR-CPA secure based on the CDH assumption in the random oracle model, given that $(\mathrm{Setup}_2, \mathrm{Extract}_2, \mathrm{Encrypt}_2, \mathrm{Decrypt}_2)$ is one-way.