Literature research: A cryptographic protocol to maintain confidentiality when storing multi-owner data

Jonas Hagg (jonas.hagg@tum.de)
Seminar Inverse Transparency (WS 21/22)
Advisor: Valentin Zieglmeier

Abstract

The construction of a cryptographic protocol to maintain confidentiality when storing multi-owner data requires the combination of two technologies: access control and cryptography. This paper is a literature research about the latter: How can data be encrypted so that multiple receivers can decrypt it.
Depending on how the set of receiving users is defined (explicitly or implicitly), different encryption techniques need to be considered. Explicit user definition refers to the notation of broadcast encryption (*BE*), which again can be implemented based on various technologies. Implicit user definition can be achieved with classical attribute-based encryption (*ABE*).
This literature review introduces and evaluates currently proposed approaches to implement these protocols. It is intended to provide an introduction into the topic and its related challenges. It also aims to support the decision of protocol designers: Which technology to choose for a practical application? Therefore, this paper elaborates a decision tree.

1 Introduction

The challenge to store encrypted data while multiple entities can access this data becomes highly relevant in the age of cloud computing [1]. While more and more users tend to use cloud storage services, the confidentiality of the uploaded data is often neglected [1]. Especially enterprises, however, can barley upload data without encrypting it [2]. Usually, many employees need to download and access the same confidential data. Hence, a cryptographic protocol is required that maintains confidentiality while multiple entities can access data.
Two requirements of such protocols are particularly relevant. First, the protocol needs to ensure confidentiality in such a way, that many users can decrypt the same encrypted data. Second, multiple users can modify the encrypted data because the protocol requires multi-owners. The second requirement can be implemented using access control techniques [3]. Access control ensures that only data owners are allowed to modify data, while others can only read the data. This is not related to the question if the accessed data is encrypted or not. Consider the case, where a storage provider enforces access control but the data is stored in plaintext. Through access control only authorized users are able to read or write the data. However, the storage provider itself has unlimited access if the data is not encrypted. On the other hand, storing encrypted data without access control mechanisms violates integrity because everybody can modify data. Thus, both requirements need to be satisfied in order to maintain confidentiality when storing multi-owner data.
Access control techniques are already well researched [3]. Thus, this paper focuses on the first requirement and provides a literature research about currently proposed cryptographic schemes which encrypt data so that multiple users can decrypt it.
In general, defining those users who are able to decrypt data can be done in two ways: Either by explicitly specifying all identities of users (e.g. $\{Alice, Bob\}$), or by implicitly defining a policy (e.g. $\{u \in U \mid \text{u works for company X}\}$). Each user which owns attributes fulfilling this policy is then allowed to decrypt. The explicit definition matches the technical notation of *broadcast encryption* (*BE*) [4]. The implicit definition can be implemented using *attribute-based encryption* (*ABE*) [5].
BE was originally proposed for broadcasting pay TV: A paying customer received a decryption box to decode a broadcasted but encrypted TV signal [6]. In the last decade, however, the concept of broadcast encryption was adopted to encrypt data for multiple receivers over the internet, e.g. by [1, 7]. Currently proposed approaches to implement broadcast encryption schemes (explicit user definition) are based on different encryption techniques: Identity-based encryption (*IBE*), certificate-based encryption (*CBE*), proxy re-encryption based encryption (*PRE*) and attribute-based encryption (*ABE*). The latter, however, directly allows implicit user definition without any additional construction.

Encrypting data which can be decrypted by many receivers is different from encrypting data for a single receiver, which is well researched in terms of classical symmetric and asymmetric cryptography [3]. However, most protocols implementing *BE* rely on these classical concepts. Moreover, broadcast encryption is not to be confused with *group encryption* as proposed in [8]. Group encryption schemes introduce of a verifying entity: This verifier checks if a given ciphertext can be decrypted by a member of the group. In particular, a ciphertext can only be decrypted by a single entity. The verifier checks if such a entity exists in the given group of users.

Contribution This literature review addresses the challenge to encrypt data which can be decrypted by multiple users. Currently proposed techniques are introduced, explained and finally evaluated in a decision tree. To better understand the differences of the underlying concepts, for each technique one publication (i.e. its algorithms and notations) is exemplary shown and explained. Moreover, for each approach the practical application of file sharing is sketched. This aims to provide an overview for the reader: What schemes are currently researched and what are their differences and similarities? The result of this analysis will be a decision tree. It might be used to make reasonable decisions about which technology to use in practice.

Organization Since *BE* can be implemented with *ABE*, there is no separate section covering implicit user definition. In fact, the analysis of *BE* schemes will cover this topic simultaneously. Thus, this paper is structured as followed: First of all, the relevant concepts needed to understand this paper are explained in Section 2. In Section 3 the technical concept of broadcast encryption is introduced. Additionally, this section also points out common terms used within the domain of broadcast encryption. Sections 4 (*IBE*), 5 (*CBE*), 6 (*PRE*) and 7 (*ABE*) then introduce and evaluate the different techniques proposed to realize *BE*. All approaches are arranged in a decision tree in Section 8. Finally, Section 9 concludes the paper.

2 Terms and Definitions

This section covers all technical terms needed to read and understand the content of the paper.

Symmetric cryptography In a symmetric cryptography scheme data is encrypted and decrypted using the same (symmetric) key. Everyone knowing this symmetric key can encrypt and decrypt data. This implies, that the symmetric key needs to be negotiated securely between the sender and receiver before encrypted data can be exchanged. This requires some kind of secure channel for key negotiation. Symmetric encryption schemes are highly optimized in hardware and very efficient. Thus, they are relevant for encrypting and decrypting big amounts of data. Symmetric cryptography is a synonym for private-key cryptography. [3]

Asymmetric cryptography Each participant in an asymmetric cryptography scheme generates two keys (a.k.a. key-pair) upon entering the system: A private-key, which is kept secret and is only known by the participant itself, and a public-key, which is publicly available and known by all participants. Suppose that Alice already generated her key-pair $(priv_A, pub_A)$. If Bob wants so send encrypted data to Alice, Bob uses the public-key of Alice (pub_A) to encrypt data. This encrypted data can only be decrypted with the private-key of Alice ($priv_A$). This construction has the advantage, that there is no need to securely negotiate a secret between the sender and receiver. Since these schemes suffer from intense mathematical computations, they are not as efficient as symmetric schemes. Asymmetric cryptography is a synonym for public-key cryptography. [3]
For the later analysis of different broadcast encryption techniques the *certificate revocation problem* is relevant. To understand this problem, this paragraph shortly introduces the concepts of certificates, certificate authorities and the revocation problem as described in [3] and [9]: A major problem with asymmetric cryptography is the question of how Bob can be sure, that the public-key of Alice actually belongs to Alice. This is important for Bob, because he wants to make sure that only Alice is able to decrypt. Suppose the following scenario: An attacker pretends to be Alice and tricks Bob into encrypting a message with the public-key of the attacker instead of the public-key of Alice. After receiving the ciphertext, the attacker then decrypts the data, analyzes or modifies it, encrypts it again with the public-key of Alice and finally sends it to Alice. Neither Bob nor Alice would notice, that the attacker interfered. Such *Man-in-the-Middle* attacks can be mitigated by introducing certification authorities (*CA*), which attest that the identity of Alice belongs to her public-key. The result of this attestation is a *certificate*[1], which binds the identity of Alice to her public-key. In other words, the certificate verifies the authenticity of the public-key. Bob uses this certificate to ensure, that the public-key he uses for encryption actually belongs to Alice. Only if Bob has a certificate for Alice from a trusted *CA*, he should send encrypted data to Alice. However, such a certificate can become invalid[2]. Encrypting data without having a valid certificate can seriously affect security: Consider the case where the private-key of Alice is stolen by an attacker. All data which is encrypted using the public-key of Alice can be directly decrypted by the attacker. Thus, before

[1]A certificate is usually a cryptographically signed data structure.

[2]There are different reasons why a certificate can become invalid: For example, because a certain amount of time has passed since the certificate was issued. Alice can also manually force the invalidation of her own certificate. This could happen if the private-key of Alice was stolen.

encrypting any data, Bob always needs to verify whether the certificate is still valid. This certificate verification is a major practical problem because it always requires up-to-date information about the current certificate status. The distribution of fresh verification information requires a lot of infrastructure. In literature this is usually called the *certificate revocation problem* [9]. A potential solution is *implicit certification* [9]. This implies that Bob can unconditionally encrypt data for Alice. There is no need for Bob to verify if the certificate of Alice is valid. On the other hand, Alice needs to be *certified* to decrypt the data. Thus, she can not unconditionally decrypt data. An example of implicit certification is described later in Section 4.

Key encapsulation Key encapsulation is a method to combine the benefits of asymmetric and symmetric crypto schemes. Thus, one can achieve efficient encryption and decryption without the need of an already established secure channel. To achieve the advantages of both worlds, the symmetric key k is encrypted by the asymmetric scheme and transported securely from the sender (Alice) to the receiver (Bob). Both, Alice and Bob now know the symmetric key k. Thus, Alice can simply encrypt the actual data with a symmetric encryption scheme and key k. Since Bob also knows k, he can restore the plaintext data.
Key encapsulation refers to the fact, that a symmetric key is *encapsulated* into an encrypted message. This allows the secure transport of symmetric keys using asymmetric cryptography. Once the symmetric key is exchanged, fast symmetric encryption and decryption algorithms can be used. [3]

Key escrow According to [9], key escrow refers to the fact, that a trusted entity knows the secret key of all users in the system. This happens if a trust center computes the secret keys on behalf of the users. A malicious or attacked trust center can easily decrypt all messages if a protocol suffers from key escrow. This compromises confidentiality and integrity of the data. To securely transport the generated secret keys a secure channel is required.
Key escrow can also be beneficial in an enterprise context, e.g. to provide decryption keys during vacation substitutions.

3 Broadcast encryption

The notation of broadcast encryption (*BE*) was first introduced by Fiat and Naor [4]. Since then many different approaches of broadcast encryption schemes were proposed. All of them solve the following challenge: How to broadcast encrypted data while only a explicitly defined set of users can decrypt the data? This section introduces the technical concepts of broadcast encryption in general. Moreover, it is discussed how broadcast encryption schemes can be classified and distinguished from each other.

3.1 Technical concepts

While in classical encryption schemes a sender encrypts data for a single receiver, broadcast encryption allows a sender to encrypt data for arbitrary $\mathbf{S} \subseteq \mathbf{U}$, where $\mathbf{U}$ is the set of all participating entities. Only entities within $\mathbf{S}$ are able to decrypt the data, while all entities in $\mathbf{R} = \mathbf{U} \setminus \mathbf{S}$ are not. $\mathbf{R}$ is said to be the set of revoked users for an encrypted message. A sender can dynamically change the set of receivers, e.g. message m_1 is encrypted for $\mathbf{S} = \{A, B\}$, but m_2 is encrypted for $\mathbf{S} = \{B, C\}$. [4]
The authors of [4] consider two trivial solutions which both lead to unwanted inefficiencies:

- Each user receives an individual key. A message is then encrypted for each user separately. Thus, only the defined set of users can decrypt the data. However, this does not allow the sender to transmit the same message to all users.
- For each possible subset of participating users a unique key is computed and shared with all members of the corresponding subset. This allows efficient decryption and transmission. However, each user needs to maintain many keys.

As a result of these inefficiencies different approaches were proposed in order to realize broadcast encryption. The following collection of terms aims to differentiate, categorize and characterize publications in the domain of broadcast encryption.

Efficiency In terms of efficiency, the following metrics are considered important for broadcast encryption schemes in literature: First, the size of the ciphertext and keys should be independent of the set of authorized users $\mathbf{S}$ [10]. Second, constant encryption and decryption times are preferred [11]. All these parameters highly depend on the underlying cryptographic principles.

Collusion resistance According to [12], a desirable property of a *BE* scheme is full collusion resistance: If the collusion of all revoked entities in $\mathbf{R}$ doesn't reveal any information about the broadcasted message, the scheme is said to be fully collusion resistant.

Asymmetric and symmetric broadcast encryption As explained in [13], one can separate between asymmetric and symmetric *BE* schemes. Within a symmetric *BE* scheme only the trusted entity of the system can encrypt and broadcast data. All other participants can only receive data. In an asymmetric scheme, however, the trusted entity not only provides key material, but also public parameters. The knowledge of this public parameters allows every entity within the system to broadcast data to a dynamically chosen set of users $\mathbf{S}$. Thus, symmetric schemes can be seen as *one-to-many* channels and asymmetric schemes as *many-to-many* channels. In this paper all analyzed systems are asymmetric *BE* schemes.

Stateful and stateless broadcast encryption In [13], the authors state the difference between stateful and stateless schemes. In a stateless broadcast encryption scheme the key material of an entity is only distributed once and does not change, even if a user joins or leaves the system. In a stateful system, however, the keys of an entity might require updates if the group of users changes. This property is only relevant if the scheme is dynamic.

Dynamic broadcast encryption According to [14], a dynamic *BE* scheme allows the group manager to invite new entities or to revoke participating entities. Thus, the set of users **U** can increase or shrink. In a stateful scheme this might require key updates for existing entities. In a stateless scheme keys don't require renewal once **U** has changed.

Trusted entity The later described techniques to achieve broadcast encryption mainly differ in key distribution and key generation. Overall, one can distinguish schemes which require a trusted entity (e.g. [15]) from schemes which do not rely on a trusted entity (e.g. [16]). A trusted entity is always a single point of failure and requires special attention in terms of security [3].
If the trusted entity computes private-keys on behalf of participating users, the protocol suffers form key escrow.

3.2 File sharing application

Figure 1 shows an exemplary application of a broadcast encryption scheme. The following sections will take up this practical application. This aims to provide a better understanding of the different approaches.
Suppose a user Alice (A) who wants to share files with Bob (B) and/or Charlie (C). To achieve this, Alice uploads encrypted files to a data center. This data can be downloaded by Bob and Charlie. Decryption, however, can only be performed by those, who were authorized during encryption. This requires the following steps:

1. A trusted entity (trust center, group manager) initializes the system with necessary parameters.
2. All participating users create cryptographic key material. The exact procedure highly depends on the underlying cryptographic scheme. This example relies on a trusted entity to compute keys. In literature there are also schemes which implement key management without the need of a trusted entity [16, 17].
3. Alice (A) uses its keys to encrypt a file for a dedicated set of users **S**.
4. The encrypted file is then uploaded to an untrusted data center.
5. Bob (B) and Charlie (C) can download the encrypted file.
6. Depending on **S**, Bob and Charlie can (not) decrypt the file.

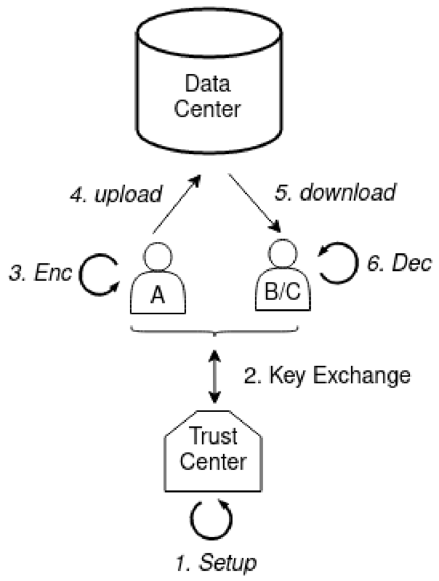

Figure 1: Conceptual procedure of a broadcast encryption scheme.

4 Identity-based broadcast encryption

Identity-based broadcast encryption schemes (*IBBE*) are a the result of merging identity-based encryption (*IBE*) with broadcast encryption techniques (*BE*) [15]. The following section describes this construction of *IBBE* schemes.

4.1 Technical concepts

IBE was initially taken into consideration by Shamir in 1985 ([18]): It is public-key cryptography, where the public-key of a user is a unique string, e.g. its email address. Private-keys are derived from this unique string via a dedicated derivation function. In order to compute private-keys, however, identity-based encryption schemes require a trusted third party that computes and distributes private-keys for each user[3].
Identity-based cryptography was later combined with broadcast encryption resulting in *IBBE* [15]. In these systems a message can be encrypted for a dedicated set of users, where each user is identified by a unique string. According to the

[3]Otherwise, everyone could compute the private-key of arbitrary users resulting in an inherent insecure public-key cryptography scheme.

initial notation of [15], they consist of four algorithms (*Setup*, *KeyExt*, *Enc* and *Dec*) and at least three entities participate: A group manager (acting as trusted third party), a sender and a receiver. The group manager is sometimes also called private-key generator (*PKG*), because it is responsible for deriving private-keys from public identities. The scheme works as followed:

1. *Setup*: The group manager initialises the system and creates a manager public-key *pkey* and a manager master key *mkey*, which is the secret key of the group manger.

   $$pkey, mkey = Setup(params)$$

2. *KeyExt*: This algorithm computes a private-key for a participating user based on its public identity. This private-key is named $skey_{id}$. Moreover, the algorithm requires *pkey* and *mkey* and thus can only be executed by the group manager. As a result, the group manager knows the private-key of each participating user (key escrow).

   $$skey_{id} = KeyExt(pkey, mkey, id)$$

3. *Enc*: This encryption algorithm takes a set of receivers **S** (identified by their public IDs) and the public-key of the group manager *pkey* and computes a *key* and a header *hdr*. The *key* is later used for encrypting data in a symmetric encryption scheme (key encapsulation). The *hdr* is public available and can be used by authorised receivers to restore the symmetric *key*.

   $$key, hdr = Enc(S, pkey)$$

   Some other *IBBE* schemes (e.g. [19] and [10]) do not encapsulate a symmetric key, but directly yield a ciphertext which can only be decrypted by authorized receivers. In this case the plaintext *M* is required as input parameter. The encryption algorithm directly computes the ciphertext *C*. Thus, the notation changes:

   $$C = Enc(S, M, pkey)$$

4. *Dec*: The decryption algorithm restores a shared key, which can be used in a subsequent step as input for a symmetric decryption algorithm. A successful decryption requires the following input parameters: The set of receivers *S*, the header *hdr* (which was the result of the encryption), the *id* with related secret key $skey_{id}$ and the public-key *pkey* of the group manager.

   $$key = Dec(S, hdr, id, skey_{id}, pkey)$$

   If the encryption algorithm does not apply key encapsulation, the decryption algorithm computes the plaintext *M* and the notation slightly changes:

   $$M = Dec(S, C, id, skey_{id}, pkey)$$

4.2 File sharing application

The concept of *IBBE* can be used to implement the file sharing application illustrated in Figure 1. *IBBE* defines how keys are handled and managed within the system. The concrete communication between a single user and the trusted entity is shown in Figure 2: Each user has a public-key (its identity *id*), which is sent to the trust center. The trust center then computes the corresponding private-key for this user. Later during encryption the user instantiates **S** with a set of authorized users. Each user in **S** can finally decrypt the files using its private-key.

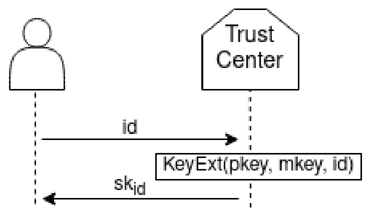

Figure 2: Key management in *IBBE*: The trust center computes the private-key for each user, based on its identity.

4.3 Evaluation

This subsection summarizes the advantages and disadvantages of *IBBE* schemes.

Advantages:

- They avoid the certificate revocation problem introduced by classical public-key encryption schemes because they make use of implicit certification (see details in Section 2): The authenticity of the public-key is attested implicitly, because a user can only participate within the system if the trusted entity has authorized the user. In particular, key material is only distributed to authenticated users. Moreover, a user can only decrypt with valid and fresh information (e.g. a secret-key) issued by the trust center. This implicitly makes sure, that only entities which were authorized by the trust center can decrypt data. [9]

- Theoretically, the trust center can be closed after all keys were distributed without affecting security [18]. In such a case, however, the system does not allow dynamically joining users.

- If the trust center stays open, the system is not only dynamic but also stateless. The key material for existing users does only depend on their identity. Thus, whenever a user joins or leaves the system no key updates are necessary.

Disadvantages:

- Since the trust center computes the secret keys for all users, these schemes suffer from key escrow [9].
- Each time a user wants to add or revoke a user from **S**, the data needs to be downloaded, decrypted, encrypted with a new set **S** and finally uploaded again [20].
- To securely transport the secret key from the trust center to a user, a secure channel is required.

5 Certificate-based broadcast encryption

Certificate-based encryption (*CBE*) was introduced by Gentry in 2003 [9]. Later, this idea was adopted with broadcast encryption by allowing multiple receivers [11, 21]. This section introduces the idea of *CBE* and describes how they can be used to construct broadcast encryption.

5.1 Technical concepts

The construction of *CBE* schemes is motivated by the following observations [9]:

- Classical public-key encryption (*PKE*) schemes suffer from the certificate revocation problem (detail in Section 2)
- Identity-based encryption schemes suffer from key escrow (details in Section 4). However, since they make use of implicit certification (details in Section 4), *IBE* schemes do not suffer from the certificate revocation problem.
- Combining both techniques – *PKE* and *IBE* – yields the new construction *CBE*. It neither suffers from key escrow nor from the certificate revocation problem.

Consider the scenario where Alice wants to send encrypted data to Bob. According to [9], *CBE* schemes function as followed[4]:

1. Alice generates a public-key-pair with a *PKE* key generation algorithm. She keeps the private-key secret. Not even the trust center is allowed to know this private-key.
2. Alice requests a certificate for her public-key. This certificate is created by the trust center by executing the *IBE* key generation algorithm. Thus, this certificate can technically also be seen as a second secret key for Alice, which is also known by the trust center.
3. Bob also executes step 1 and step 2.
4. In order to encrypt data for Bob, Alice uses Bobs public-key and encrypts the message twice: Once, with the *IBE* encryption algorithm and once with the *PKE* encryption algorithm. The doubly encrypted message is then sent to Bob. In particular, there is no need for Alice to check if Bob has a valid certificate. This avoids the certificate revocation problem.
5. Upon receiving the encrypted message, Bob needs to double decrypt the ciphertext. Once, with the *IBE* decryption algorithm and once with the *PKE* decryption algorithm. This enforces the following: 1) Only Bob can apply the *PKE* decryption, because only he knows the required private-key and 2) Bob needs a valid certificate, because otherwise he could not apply the *IBE* decryption. In particular the first fact avoids key escrow (because the trust center does not know the private-key of Bob) and the second fact avoids the certificate revocation problem (because Bob can only decrypt with a valid certificate).

This idea was later adopted to broadcast encryption, leading to certificate-based broadcast encryption. According to the notation of [11], such a scheme can be described as the following set of algorithms[5]:

1. *Setup*: Based on a security parameter k, this algorithm outputs a system master key MK (kept secret) and system parameters $params$, which are public. This algorithm is executed by the trust center.

$$(MK, params) = Setup(k)$$

2. KeyGen: This key generation algorithm generates a key-pair based on the system parameters and the identity of the user ID_i. This algorithm is executed by a user. It is the equivalent to the *PKE* key generation algorithm explained above.

$$(PK_i, SK_i) = KeyGen(params, ID_i)$$

3. CertGen: This certification algorithm issues a certificate $Cert_i$ for the requesting user. As input it requires the system parameters $params$, the system master key MK, the user identity ID_i and the public-key of the user PK_i. Thus, this algorithm is executed by the trust center. It is the equivalent to the *IBE* key generation algorithm explained above.

$$Cert_i = CertGen(params, MK, ID_i, PK_i)$$

4. *Encrypt*: This encryption algorithm takes the system parameters $params$, a plaintext M and the set of target users S as input. It then computes a ciphertext CT. This

[4]For the sake of reduced complexity this is a simplified description.

[5]This notation does not separate strictly between *IBE* and *PKE* algorithms as described before.

ciphertext can only be decrypted by users in S, which have a valid certificate.

$$CT = Encrypt(params, S, M)$$

5. *Dec*: The decryption algorithm restores the plaintext M from the cipher if the provided identity ID_i is in S, if the certificate $Cert_i$ is valid and if the secret key SK_i belongs to ID_i.

$$M = Dec(params, CT, ID_i, SK_i, Cert_i)$$

5.2 File sharing application

To illustrate the file sharing example via *CBE* based broadcast encryption, again consider Figure 1. Before Alice can encrypt files she needs to generate her key-pair. Strictly speaking, Alice does not even need to receive a certificate from the trust center, because this is only required for decrypting data. Bob, however, requires a key-pair and a certificate. The concrete key management in such a system can be seen in Figure 3: Once a user created its public-key-pair based on its *ID*, the trust center issues a certificate for this user. With the help of this certificate the user can later decrypt data.

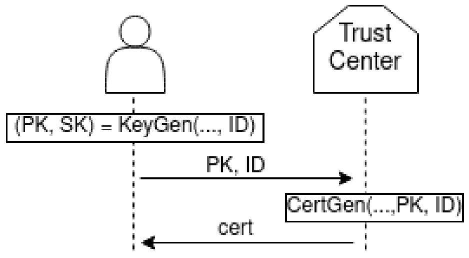

Figure 3: Key management in *CBBE*: The user computes its own key-pair. The trust center issues a certificate for the user, which is required for decrypting data.

5.3 Evaluation

This subsection summarizes the advantages and disadvantages of *CBE* based broadcast encryption schemes.

Advantages:

- Theses system neither suffer from key escrow nor from the certificate revocation problem [9].
- The scheme proposed by [11] is implemented to be dynamic and stateless.

Disadvantages:

- A secure channel to transport the certificate from the trust center to the user is required. Even though key escrow can be avoided, the trust center is still a single point of failure.
- Each time a user wants to add or revoke a user from **S**, the data needs to be downloaded, decrypted, encrypted with a new set **S** and finally uploaded again.

6 Proxy re-encryption based broadcast encryption

Broadcast encryption schemes can be implemented with a technique called proxy re-encryption (*PRE*). This section introduces the concepts of such encryption schemes.

6.1 Technical concepts

Proxy re-encryption systems are cryptographic schemes, in which a third party (a.k.a. a *proxy*) converts a ciphertext, which was originally intended for Alice, to a new ciphertext, which is intended for Bob. For this conversion the proxy requires a re-encryption key. Usually these systems ensure, that the proxy does not obtain knowledge about the plaintext during re-encryption. Nevertheless, this proxy needs to be semi-trusted. [22]
Similar to the example given in [22], consider a user Alice, who stores encrypted data $\mathbf{E}_A$ in the cloud. $\mathbf{E}_A$ can only be decrypted by Alice. Suppose that the data center is also the re-encryption proxy. In order to delegate this data to Bob, Alice creates a re-encryption key $rk_{Alice,Bob}$. This key is transferred to the proxy, which re-encrypts $\mathbf{E_A}$. The result of this re-encryption is $\mathbf{E}_B$. Bob can then download and decrypt $\mathbf{E}_B$. Alice benefits from reduced computation costs, because there is no need to download, re-encrypt and upload the data once she wants to share $\mathbf{E}_A$ [20]. Moreover, the network load is relieved compared to plain broadcast encryption schemes, since less data is transmitted [22].
Proxy re-encryption systems can be distinguished into *unidirectional* and *bidirectional*: In an unidirectional scheme Alice uses an re-encryption key to delegate data to Bob, but Bob can not use the same key to also delegate data to Alice. In an bidirectional scheme, however, the same re-encryption key can be used by Bob for delegating data to Alice. [16]
Classical proxy re-encryption schemes do not require a trusted entity to distribute keys [16, 17]. However, there are also techniques which require a trusted entity for the purpose of key distribution [23].
According to the notation of [17], a proxy re-encryption scheme can be described as the following set of algorithms:

1. *Setup*: Based on a security parameter k, this algorithm outputs public parameters which are shared between all participants of the system.

$$params = Setup(k)$$

2. *KeyGen*: This key generation algorithm generates a asymmetric key-pair for a participating entity i. It does

not require a trusted entity but can be executed by the user itself.

$$(pk_i, sk_i) = KeyGen()$$

3. *ReKeyGen*: This re-encryption key generation algorithm generates a re-encryption key. This key allows the proxy to convert a ciphertext of user i to a ciphertext of user j. The key is computed by entity i. To achieve this, the algorithm requires the secret key of user i and the public-key of user j:

$$rk_{i,j} = ReKeyGen(sk_i, pk_j)$$

4. *Enc*: This encryption algorithm takes a plaintext M and a public-key pk_i as input and outputs a ciphertext C_i, which can only be decrypted with secret key sk_i.

$$C_i = Enc(pk_i, M)$$

5. *ReEnc*: This is the re-encryption algorithm. It takes a re-encryption key $rk_{i,j}$ and a ciphertext C_i as input and outputs a re-encrypted ciphertext C_j, which can only be decrypted with secret key sk_j.

$$C_j = ReEnc(rk_{i,j}, C_i)$$

6. *Decrypt*: This decryption algorithm takes a cipher C_i and a secret key sk_i as input. It outputs the plaintext M if the cipher is encrypted for user i.

$$M = Dec(sk_i, C_i)$$

6.2 File sharing application

In Figure 4 the procedure of a proxy re-encryption based broadcast encryption system is illustrated. In this case we consider a file sharing application. The proxy is part of the data center, where the encrypted files are stored. The steps are:

1. Computation of a key-pair by each user.
2. Files are encrypted by Alice using her own public-key. Thus, this data can only be decrypted by Alice.
3. Alice uploads the encrypted files are uploaded to the data center.
4. Suppose Alice wants to share the uploaded data with Bob. To achieve this, Alice needs to compute a re-encryption key $rk_{Alice,Bob}$. This key is then sent to the proxy, which applies the re-encryption algorithm to the encrypted files.
5. Bob requests and downloads the encrypted files.
6. Bob decrypts the files using his secret key.

Figure 5 further details how keys are managed in classical proxy re-encryption based schemes: No trusted entity is required. Each user itself computes 1) its own key-pair and 2) re-encryption keys if required.

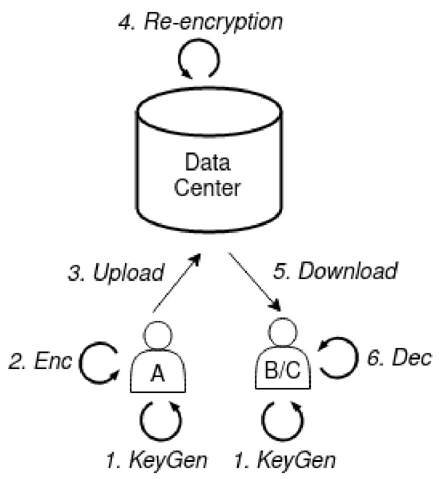

Figure 4: Conceptual procedure of a proxy re-encryption based broadcast encryption scheme.

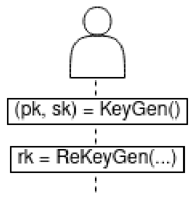

Figure 5: Key management in a proxy re-encryption based broadcast encryption scheme: No trust center is required. Each user computes its key-pair. Re-encryption keys are compute by a user when necessary.

6.3 Evaluation

This subsection summarizes the advantages and disadvantages of *PRE* based broadcast encryption schemes. Advantages:

- Since a user only needs to provide a re-encryption key, there is only minimal overhead if data needs to be shared with a new user. All heavy cryptographic tasks are shifted to the proxy. [20]
- As shown by [17], *PRE* based broadcast encryption can be implemented to be dynamic and stateless: Each entity is responsible for creating its own keys. Dynamically joining or leaving users do not require updates for already established keys.
- The set of authorizes users **S** is not required during initial encryption. A user is added to **S** by providing a re-encryption key to the proxy.

Disadvantages:

- There are two approaches to implement these systems: with a trusted entity and without. Introducing a trusted entity usually also introduces key escrow while avoiding a trusted entity often results in the certificate revocation problem [9].
- Once the proxy knows a re-encryption key $rk_{Alice,Bob}$, all confidential data of Alice can be re-encrypted for Bob. Thus, the proxy needs to be semi-trusted [16].

7 Attribute-based encryption

All protocols in the previous sections rely on explicit user definition. Section 7.1 technically introduces attribute-based encryption (*ABE*), which provides implicit user definition by design. At the same time, *ABE* can also be used to construct *BE* schemes [24]. This will be covered in Section 7.2.

7.1 Technical concepts

According to [25], the following points elaborate the main features of classical attribute-based encryption:

- Attribute-based encryption is a approach to provide access control within the domain of cryptography. Without attribute-based encryption, access control techniques need to be implemented by the server. Access will be granted if the requesting user is allowed to read/write the data[6]. Thus, the server ensures that only authorized entities access data. Consider the case where this server is compromised: The attacker has unlimited access to the stored data because the access control techniques are no longer in place and the data itself is not encrypted.
- Attribute-based encryption schemes shift the logic of access control into encryption and decryption algorithms. A distinction is made between a set of possible attributes (e.g. A,B,C) and a policy. Policies are logical expressions over attributes (e.g. $A \wedge B \wedge \neg C$). A private-key of an user contains certain attributes. During encryption, a user specifies a policy which is encoded into the ciphertext. A ciphertext can then only be decrypted with a private-key, which fulfills that policy.
- Within the domain of *ABE* one can differ between *ciphertext-policy ABE* and *key-policy ABE*. The above described scenario, where the policy is encoded into the ciphertext and the key of the user is checked against this policy is called *ciphertext-policy ABE*. On the contrary, *key-policy ABE* describes the scenario, where the policy is encoded into the key of the user. Thus, a user can only decrypt a ciphertext which is annotated with the corresponding attributes.

The notation of *ciphertext-policy ABE* directly can be used to encrypt data for multiple receivers. It relies on implicit user definition. During encryption a policy is defined. All users which are equipped with attributes fulfilling this policy can decrypt data. While the initially proposed schemes rely on a trust entity [5], there are also decentralized approaches that avoid key escrow [26, 27].

7.2 Attribute-based broadcast encryption

The technique of attribute-based encryption can be used to implement broadcast encryption. This can be achieved by a scheme, which encodes the set of receiving users **S** into the ciphertext. A user can only decrypt this ciphertext if it is a member of **S**. [24]
According to the notation of [24], an *ABE* based broadcast encryption scheme can be described as the following set of algorithms:

1. *Setup*: Based on a security parameter k, this algorithm outputs a master public-key mpk and a master secret key msk. This is executed by a trusted entity, which shares mpk with all participants.

$$(mpk, msk) = Setup(k)$$

2. KeyGen: This key generation algorithm generates a secret key sk_Y based on the master public-key, master secret key and Y, which identifies the user requesting the key. In [24] this is simply an integer.

$$(sk_Y) = KeyGen(mpk, msk, Y)$$

3. *Enc*: This encryption algorithm computes a ciphertext ct. It takes a plaintext μ, the master public-key mpk and X

[6] E.g. if the user works for a certain company.

as input. X identifies an arbitrary subset of participating users. In [24] this is simply a set of integers.

$$ct = Enc(mpk, X, \mu)$$

4. *Dec*: This decryption algorithm takes the master public-key mpk, a cipher ct and a secret key s_Y as input. If Y matches the policy encoded in ct the algorithm yields the plaintext μ, otherwise it returns $\perp$.

$$\mu = Dec(mpk, ct, sk_Y)$$

7.2.1 File sharing application

Again have a look to Figure 1. The file sharing application illustrated there can be implemented with *ABE* techniques. *ABE* defines the distribution of keys within the system. This is depicted in Figure 6: The trust center is responsible for computing secret keys for each user. During encryption Alice then defines the set of users **S**, which is internally interpreted as a policy. During decryption, only users with a key satisfying this policy (i.e. all users in **S**) can decrypt the data.
However, keep in mind, that this is a very specific case of *ABE*. In particular, one could also use a scheme without the need of explicitly defining all target users **S**. To achieve this, one could rely on generic policies and attributes to encode a custom and fine-grained access control scheme [24]. In such a scheme, it is not crucial whether a user is within **S** or is not. A user can decrypt data if and only if the user matches the required policy: For example, if the user works in department A and is not involved in project X.

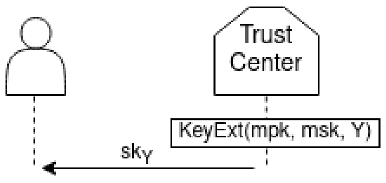

Figure 6: Key management in an attribute-based broadcast encryption scheme: The trust center computes the secret key for each user.

7.2.2 Evaluation

This subsection summarizes the advantages and disadvantages of *ABE* based broadcast encryption schemes. Advantages:

- *ABE* introduces efficient broadcast encryption schemes with small system parameters [24].
- As shown by [24], such systems can be implemented to be dynamic and stateless, since no key updates are necessary if a user joins or leaves the system.

Disadvantages:

- Each time a user wants to add or revoke a user from **S**, the data needs to be downloaded, decrypted, encrypted with a new set **S** and finally uploaded again. However, this might be avoided by using a more generic *ABE* approach: Rather than defining the set of target users **S** a generic policy could be defined. All users fulfilling this policy can decrypt data.
- Classical *ABE* schemes suffer from key escrow because they rely on a trust center that distributes secret keys. There are more recent approaches to avoid this [26, 27]. Unfortunately, these key-escrow free schemes do not consider the special case of attribute-based broadcast encryption.

8 Decision tree

All introduced schemes in this paper have certain benefits and limitations. This section arranges all discussed approaches in a decision tree (Figure 7). It is intended to provide support for a decision, where a protocol designer needs to choose from one of these technologies.
The first decision to take is whether the protocol should abstract from identities. If users need be defined implicitly, *ABE* would be a good choice. However, if explicit user definition is required, a *BE* scheme should be implemented. The consequences of this decision are discussed in Section 8.1. Moving forward within the decision tree, a designer needs to choose the underlying technology if the protocol is based on *BE*. Their differences are summarized and discussed in Section 8.2.

8.1 Explicit or implicit user definition?

If the receiving set of users is defined explicitly, the encrypting entity needs to provide all users which are able to decrypt. For example $S = \{Alice, Bob\}$. As a result, the data can only be decrypted with the private-key of one of these users. The advantage of this approach is the clearness of who can access the data. However, the set of receiving users is determined at encryption time.
On the other hand, the set of users can also be implemented implicitly. This is achieved by creating a policy. For example $S = \{u \in U \mid \text{u works for company X AND u is older than 30}\}$. This abstracts from identities and only allows decryption if a user has attributes fulfilling the policy. During policy creation, it is not clear how many user can decrypt the data. Dynamically joining and leaving users are easy to maintain. However, each user needs to be equipped with attributes.

8.2 How to implement broadcast encryption?

Broadcast encryption schemes can be implemented using different encryption techniques: Identity-based encryp-

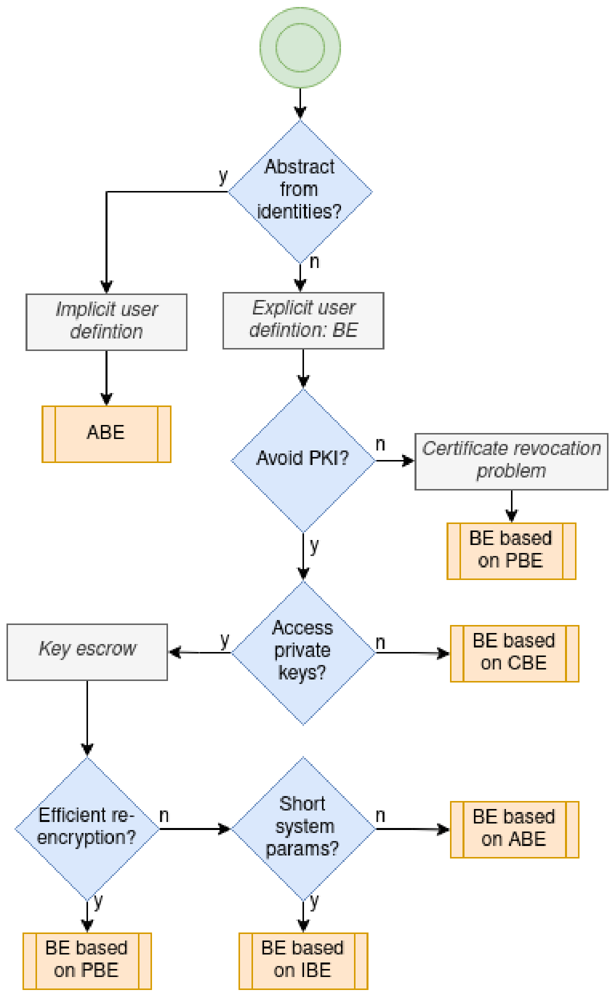

Figure 7: Decision tree: What encryption technology to choose for a protocol that maintains confidentiality when storing multi-owner data?

tion, proxy re-encryption, attribute-based encryption and certificate-based encryption. Table 1 visualizes the major achievements and limitations of broadcast encryption schemes.

All researched techniques are capable of implementing dynamic and stateless broadcast encryption systems. However, none one of them can renounce a trusted entity. *IBE* and *ABE* based schemes suffer from key-escrow because their trusted entity distributes private-keys to users. *CBE* prevents key-escrow because it applies double encryption, but a trusted entity for certificate distribution is still required. The presented *PRE* based scheme does not require key distribution because each user generates its own key material. However, the re-encryption proxy also needs to be a semi-trusted third party.

All approaches require a secure channel. *IBE*, *ABE* and *CBE* based approaches require it to securely transport private-keys or certificates. *PRE* based schemes require a secure channel to securely transport re-encryption keys to the proxy.

The use of a trusted entity to distribute private-keys eliminates the problem of the certificate revocation problem because they benefit from implicit certification. Thus, *IBE*, *ABE* and *CBE* based schemes do not suffer from this problem. For *PRE* based schemes this question depends on the decision whether to use a trust center or not.

IBE and *ABE* based approaches suffer from key escrow because they require a trusted entity for key distribution. *CBE* prevents this problem by double encryption. For *PRE* based schemes key-escrow might become a problem if a trust center is introduced.

The last row in Table 1 visualizes how efficient the set of authorized user **S** can be changed for an existing ciphertext. This can be done efficiently by *PRE* based schemes because a user only needs to send the re-encryption key to the proxy. All heavy cryptographic tasks are outsource and performed by the proxy. In *IBE* and *CBE* based approaches the user needs to download, decrypt, re-encrypt and upload the cipher to modify **S**. The same holds for *ABE* based schemes.

This rather abstract analysis is depicted more graphically and solution-oriented in the decision tree (Figure 7). Avoiding a *PKI* eliminates the certificate revocation problem. If nobody should be able to access private keys of participating users, then *CBE* is a good choice. Otherwise, the scheme suffers from key-escrow. *PRE* schemes are tailored specifically to the problem of broadcast encryption. The usage of a proxy reduces computation costs for participants resulting in efficient re-encryption. *ABE* based schemes benefit from short system parameters. *IBE* is not used exclusively for broadcast encryption, but it can be found in various other applications [28, 9]. Thus, *IBE* based approaches benefit from a lot of research and security analysis.

Table 1: Overview of introduced *BE* schemes with their benefits and limitations.[7]

BE based on	IBE	PRE	ABE	CBE
dynamic	✓	✓	✓	✓
stateless	✓	✓	✓	✓
avoid trusted entity	x	*x*	x	x
avoid secure channel	x	x	x	x
avoid certificate revocation problem	✓	∼	✓	✓
avoid key escrow	x	∼	x	✓
efficiently change set of authorized users **S**	x	✓	x	x

9 Conclusion

This closing section summarizes the most important results of the paper. It shows that maintaining confidentiality when storing multi-owner data can be realized using various technologies. Each approach comes with advantages and disadvantages. For details have a look to the evaluation sections for each technique above. There is probably no optimal all-fit-one solution. Thus, the approach to choose in a practical application is highly context dependent. To help a designer to choose one of these technologies, a decision tree was elaborated (see Section 8).
Classical *ABE* provides a lot flexibility because of its policy-based approach. It can be understood as implicit user definition[8]. If the set of receiving users needs to be defined explicitly, broadcast encryption schemes become relevant: *PRE* based schemes enable the most efficient broadcast encryption. *ABE* based broadcast encryption schemes benefit from short system parameters. *IBE* is used in other domains; thus it is a well researched field of cryptography. *CBE* finally improves *IBE* by preventing key escrow.

9.1 Limitations

Four fundamental approaches to implement broadcast encryption are considered in this paper. Due to submission and complexity requirements, this paper did not consider approaches, where the different techniques are combined: For example, there are publications which combine *IBE* with *PRE* ([23]). Moreover, many different schemes are proposed based on each technique. There is a wide range of goals they try to achieve. For example, some publications consider different security notations or they try to build more efficient, quantum-secure or anonymous schemes. Subsequent research could investigate literature research for each technique.
As explained in the first section, a protocol that maintains confidentiality while storing multi-owner data also requires access control. Combining access control with the proposed encryption techniques to construct a consistent and secure protocol can be subject of future work.

[7] Please note, that this is not an absolute result. Some characteristics depend on the concrete design of a scheme. For example, *PRE* based broadcast encryption schemes can be implemented with or without a trusted entity responsible for distributing keys. Thus, some characteristics can not be decided on the level of the used encryption technique. This is indicated by orange cells annotated with '∼'. For details about each technique have a look in the corresponding section.

[8] The same idea is followed by role-based access models, where access is not given based on the identity of a user. Access is permitted or rejected based on *roles* an identity owns. This allows to dynamically change the role of users and to implement hierarchical structures. [3]

References

[1] Liqing Chen, Jiguo Li, Yang Lu, and Yichen Zhang. Adaptively secure certificate-based broadcast encryption and its application to cloud storage service. *Information Sciences*, 538:273–289, 2020. URL https://doi.org/10.1016/j.ins.2020.05.092.

[2] Malek Najib Omar, Mazleena Salleh, and Majid Bakhtiari. Biometric encryption to enhance confidentiality in Cloud computing. In *2014 International Symposium on Biometrics and Security Technologies (ISBAST)*, pages 45–50, 2014. URL https://doi.org/10.1109/ISBAST.2014.7013092.

[3] Claudia Eckert. *IT-Sicherheit*. Oldenbourg Wissenschaftsverlag, 2013. URL https://doi.org/10.1524/9783486735871.

[4] Amos Fiat and Moni Naor. Broadcast encryption. In *Annual International Cryptology Conference*, pages 480–491. Springer, 1993. URL https://doi.org/10.1007/3-540-48329-2_40.

[5] Amit Sahai, Brent Waters, and Steve Lu. Attribute-Based Encryption. In *IDENTITY-BASED CRYPTOGRAPHY*, volume 2 of *Cryptology and Information Security Series*, pages 156–168. IOS PRESS, 2009. URL https://doi.org/10.3233/978-1-58603-947-9-156.

[6] N. Kogan, Y. Shavitt, and A. Wool. A practical revocation scheme for broadcast encryption using smart cards. In *Symposium on Security and Privacy*, pages 225–235, 2003. URL https://doi.org/10.1145/1178618.1178622.

[7] Jing yi Fu, Qin long Huang, Zhao feng Ma, and Yi xian Yang. Secure personal data sharing in cloud computing using attribute-based broadcast encryption. *The Journal of China Universities of Posts and Telecommunications*, 21(6):45–77, 2014. URL https://doi.org/10.1016/S1005-8885(14)60344-7.

[8] Aggelos Kiayias, Yiannis Tsiounis, and Moti Yung. Group encryption. In *International Conference on the Theory and Application of Cryptology and Information Security*, pages 181–199. Springer, 2007. URL https://doi.org/10.1007/978-3-540-76900-2_11.

[9] Craig Gentry. Certificate-Based Encryption and the Certificate Revocation Problem. In *Advances in Cryptology — EUROCRYPT 2003*, pages 272–293. Springer Berlin Heidelberg, 2003. URL https://doi.org/10.1007/3-540-39200-9_17.

[10] Kai He, Jian Weng, Jia-Nan Liu, Joseph K. Liu, Wei Liu, and Robert H. Deng. Anonymous Identity-Based Broadcast Encryption with Chosen-Ciphertext Security. In *Proceedings of the 11th ACM on Asia Conference on Computer and Communications Security*, pages 247–255. Association for Computing Machinery, 2016. URL https://doi.org/10.1145/2897845.2897879.

[11] Jiguo Li, Liqing Chen, Yang Lu, and Yichen Zhang. Anonymous certificate-based broadcast encryption with constant decryption cost. *Information Sciences*, pages 110–127, 2018. URL https://doi.org/10.1016/j.ins.2018.04.067.

[12] Cécile Delerablée. Identity-Based Broadcast Encryption with Constant Size Ciphertexts and Private Keys. In *Advances in Cryptology – ASIACRYPT 2007*, pages 200–215. Springer Berlin Heidelberg, 2007. URL https://doi.org/10.1007/978-3-540-76900-2_12.

[13] Yevgeniy Dodis and Nelly Fazio. Public Key Broadcast Encryption for Stateless Receivers. In *Digital Rights Management*, pages 61–80. Springer Berlin Heidelberg, 2003. URL https://doi.org/10.1007/978-3-540-44993-5_5.

[14] Cécile Delerablée, Pascal Paillier, and David Pointcheval. Fully Collusion Secure Dynamic Broadcast Encryption with Constant-Size Ciphertexts or Decryption Keys. In *Pairing-Based Cryptography – Pairing 2007*, pages 39–59. Springer Berlin Heidelberg, 2007. URL https://doi.org/10.1007/978-3-540-73489-5_4.

[15] Ryuichi Sakai and Jun Furukawa. Identity- Based Broadcast Encryption. *Cryptology ePrint Archive*, 2007:217, 2007. URL https://ia.cr/2007/217.

[16] Giuseppe Ateniese, Kevin Fu, Matthew Green, and Susan Hohenberger. Improved Proxy Re-Encryption Schemes with Applications to Secure Distributed Storage. *ACM Trans. Inf. Syst. Secur.*, 9:1–30, 2006. URL https://doi.org/10.1145/1127345.1127346.

[17] Sherman S. M. Chow, Jian Weng, Yanjiang Yang, and Robert H. Deng. Efficient Unidirectional Proxy Re-Encryption. In *Progress in Cryptology – AFRICACRYPT 2010*, pages 316–332. Springer Berlin Heidelberg, 2010. URL https://doi.org/10.1007/978-3-642-12678-9_19.

[18] Adi Shamir. Identity-Based Cryptosystems and Signature Schemes. In *Advances in Cryptology*, pages 47–53. Springer Berlin Heidelberg, 1985. URL https://doi.org/10.1007/3-540-39568-7_5.

[19] Jianchang Lai, Yi Mu, Fuchun Guo, Willy Susilo, and Rongmao Chen. Anonymous Identity-Based Broadcast Encryption with Revocation for File Sharing. In *Information Security and Privacy*, pages 223–239. Springer International Publishing, 2016. URL https://doi.org/10.1007/978-3-319-40367-0_14.

[20] Linmei Jiang and Donghui Guo. Dynamic Encrypted Data Sharing Scheme Based on Conditional Proxy Broadcast Re-Encryption for Cloud Storage. *IEEE Access*, 5:13336–13345, 2017. URL https://doi.org/10.1109/ACCESS.2017.2726584.

[21] Chun-I Fan, Pei-Jen Tsai, Jheng-Jia Huang, and Wen-Tsuen Chen. Anonymous Multi-receiver Certificate-Based Encryption. In *International Conference on Cyber-Enabled Distributed Computing and Knowledge Discovery*, pages 19–26, 2013. URL https://doi.org/10.1109/CyberC.2013.13.

[22] Wei-Hao Chen, Chun-I Fan, and Yi-Fan Tseng. Efficient Key-Aggregate Proxy Re-Encryption for Secure Data Sharing in Clouds. In *IEEE Conference on Dependable and Secure Computing (DSC)*, pages 1–4, 2018. URL https://doi.org/10.1109/DESEC.2018.8625149.

[23] Matthew Green and Giuseppe Ateniese. Identity-Based Proxy Re-encryption. In *Applied Cryptography and Network Security*, pages 288–306. Springer Berlin Heidelberg, 2007. URL https://doi.org/10.1007/978-3-540-72738-5_19.

[24] Shweta Agrawal and Shota Yamada. Optimal Broadcast Encryption from Pairings and LWE. *Cryptology ePrint Archive*, 2020:228, 2020. URL https://ia.cr/2020/228.

[25] John Bethencourt, Amit Sahai, and Brent Waters. Ciphertext-Policy Attribute-Based Encryption. In *IEEE Symposium on Security and Privacy*, pages 321–334, 2007. URL https://doi.org/10.1109/SP.2007.11.

[26] Nyamsuren Vaanchig, Hu Xiong, Wei Chen, and Zhiguang Qin. Achieving Collaborative Cloud Data

Storage by Key-Escrow-Free Multi-Authority CP-ABE Scheme with Dual-Revocation. *International Journal of Network Security*, 20:95–109, 2018. URL https://doi.org/10.6633/IJNS.201801.20(1).11.

[27] Jianwei Chen and Huadong Ma. Efficient decentralized attribute-based access control for cloud storage with user revocation. In *2014 IEEE International Conference on Communications (ICC)*, pages 3782–3787, 2014. URL https://doi.org/10.1109/ICC.2014.6883910.

[28] Sanjit Chatterjee and Palash Sarkar. *Identity-based encryption*. Springer Science & Business Media, 2011. URL https://doi.org/10.1007/978-1-4419-9383-0.