Radboud University Nijmegen

Master Thesis

Decentralized Attribute-Based Encryption for DECODE

Author:
Marloes Venema
(s4126173)

Supervisor:
Dr. Jaap-Henk Hoepman

Second reader:
Dr. Peter Schwabe

in the

Digital Security Group
Institute for Computing and Information Sciences

August 17, 2018

Abstract

In ecosystems such as DECODE, measures to implement access control in a decentralized setup are paramount. One of the cryptographic tools that can be used to realize this is attribute-based encryption. This is a type of public-key cryptography which associates the secret keys with attributes rather than with the identity of the user. Because there is a multitude of schemes with several different properties, it can be hard to find the right scheme for some chosen setting. In this work, we investigate which schemes are out there and which properties they satisfy. Then we use this research to find the best attribute-based encryption scheme for DECODE and similar ecosystems. One of the disadvantages of attribute-based encryption is the size of the ciphertexts, which tends to be linear in the number of attributes. Because in the literature, the amount of schemes with constant-size ciphertexts is very small, and there are, to the best of our knowledge, no constant-size schemes in the decentralized setting that allow for expressive access policies, we will also outline a (very inefficient) decentralized multi-authority ciphertext-policy attribute-based encryption scheme with constant-size ciphertexts.

Contents

Chapter 1

Introduction

In the last decades, technology has rapidly progressed to the point where computers, other devices and the internet have become a significantly large part of our lives. In light of this development, much of our personal data have been digitized, and much of our activity on the internet is stored in databases. Companies such as Facebook and Twitter, but also every other company that provides us with services in exchange for our personal data, base their business model on selling our personal data.

Because there are so many organizations that store and process our personal data, some people feel as though they have lost control over whom gets to process which information for which purposes. Projects such as IRMA[1] and DECODE[2] attempt to give some of that control back, without individuals having to opt out of using services that require the sharing of personal data altogether.

One of the measures that can be taken is the implementation a form of access control that allows individuals to determine which organizations and other individuals get to access which data based on the attributes that the individual or organization possesses. An effective way to implement this type of attribute-based access control is to use a cryptographic primitive called *attribute-based encryption (ABE)*, which allows for fine-grained access control on a cryptographic level.

1.1 The DECODE project

The DECODE project, which is an acronym for DEcentralized Citizen-Owned Data Ecosystems, addresses these concerns by aiming to develop technology that allows individuals to control their own private data by giving them the ability to decide how it is shared with service providers.

The goals of DECODE are [Dan+17]:

- allowing participants to manage access to their private data, by granting and revoking access through entitlements;
- allowing operators to write smart rules, sign them and get the authorization to run them on DECODE;
- allowing smart rules to access private data based on entitlements and matched attributes;
- allowing everyone to record entitlements on a distributed ledger whose integrity is resilient and verifiable.

[1] I Reveal My Attributes (IRMA): `https://privacybydesign.foundation/en/`.
[2] DEcentralized Citizen-Owned Data Ecosystems (DECODE): `https://decodeproject.eu/`

DECODE aims to accomplish these goals by exploiting existing primitives such as distributed ledgers and attribute-based credentials. Moreover, DECODE does this in a decentralized fashion: distributing the trust across multiple parties such that several parties have to be corrupted in order to break any security guarantees (e.g. integrity, availability, confidentiality) of the system.

1.1.1 Access control

As we already briefly mentioned, one of the measures that can be taken to allow participants to manage access to their private data is the implementation of access control. There are several security principles that are important in this setting, among of which are confidentiality, integrity and availability. In order to guarantee all of these security principles in a decentralized manner, DECODE uses distributed ledgers (to ensure integrity and availability) and attribute-based credentials and encryption (to ensure authenticity and confidentiality).

The idea of access control is that, through entitlements, access is granted and revoked by the participant. In traditional systems, access control is often implemented in the code of the system, and it either grants or denies access to data based on the credentials (e.g. role or attribute-based) of a user. We simply have to 'trust' that this is done correctly, and that there are no participants or other parties in the system that can read the data that was not meant for them to be accessed.

In a way, there is one 'central authority' in this setup that enforces access control, which is what DECODE wants to avoid. In DECODE, the authorization of users and enforcement of access control are supposed to be done in a decentralized setting, namely by exploiting cryptographic primitives such as attribute-based credentials and encryption respectively, which both allow decentralized setups.

1.1.2 Distributed ledger

The notion of distributed ledgers is one of the core foundations of DECODE, which provides both integrity and availability (but not confidentiality). The idea of distributed ledgers is that they allow participants to perform transactions and record a verifiable proof of that transaction without having to use data within the transaction itself. Because is does not contain private data, the proof can be stored anywhere. In DECODE, the proofs of these transactions will be stored on a network of nodes. Because anyone can join the network and host a node, we do not have to put trust in one central authority.

1.1.3 Attribute-based credentials

Attribute-based credentials (ABCs) are a cryptographic primitive that allow for issuance of credentials to users in a verifiable but privacy-friendly manner: an issuer will 'sign' the data on the credentials such that other parties can verify whether the data that is stored on the credential is authentic. For this simple reason, they provide us with an authorization mechanism. On the credentials, attributes that are related to the identity of a participant are stored. These attributes will be signed and can be verified, and are therefore suited to be evaluated against access policies. Because of this reason, it may be possible to combine ABCs and attribute-based encryption such that ABCs can be used to prove possession of the attributes, and hence therefore provides the authentication part on the user's side.

1.1.4 Attribute-based encryption

Whereas ABCs are used for authentication and proving possession of attributes, attribute-based encryption (ABE) can be used as a means to enforce access control on the private data of a participant, i.e. the participant encrypts the private data under an access policy such that other participants in the system can only decrypt the data if they possess a set of attributes (and their associated keys) that satisfies the policy. This means that ABE provides us with a way to implement attribute-based access control on a cryptographic level. One of the features in ABE is that the setup can be decentralized: instead of allowing one authority to control all of the keys associated with attributes, multiple authorities distribute the keys associated with their own unique set of attributes to users that possess those attributes. The idea is that the user can authenticate towards the authority by showing his attribute-based credential, and therefore prove possession of the associated attribute. However, how this is done is not entirely trivial, and it is therefore one of the key challenges of the DECODE project. [AB+17]

1.2 Our goal and approach

Because a lot of research and literature is published on attribute-based encryption, there are various different schemes with a multitude of different properties circulating the research community. For different use cases, it is desirable to have different types of attribute-based encryption that fit the specific requirements that a system imposes. For instance, in a hospital it might be beneficial to have a central setup, whereas in large systems such as DECODE it is desirable to decentralize this setup. There might also be systems in which we require the systematic update of the keys, because the associated attributes are rather volatile. Some systems have a small set of users, and some systems have a large set of users, which means that the ABE scheme should be scalable to such large systems.

Our goal is to investigate which properties such ABE schemes may have and which are suitable for DECODE in particular. Then we will investigate which ABE schemes are already out there and which of the aforementioned properties they satisfy. Then we can compare all of them and consider which would suit the DECODE setup the best, based on the properties that we deemed suitable (see Section 3.11).

1.3 Overview

In this work, we will mainly focus on our goal, which is studying ABE, their properties and the existing schemes. Before we can describe ABE, we will give some preliminaries in Chapter 2. After that, we will focus on ABE and its properties in Chapter 3, which we will end with a list of properties that we deem suitable for DECODE. In Chapter 4, we will look at two existing schemes and compare a larger set of forty-two ABE schemes, of which we will choose the best candidate for DECODE based on the properties we specified in Section 3.11.

Because almost all of the existing ABE schemes have the unfortunate feature that ciphertexts are linear in the size of the access policy, we will focus on the use of ABE schemes with constant-size ciphertexts in the appendix. There is one particular ABE scheme that was created within a framework that aims to simplify the design and analysis of ABE schemes with the use of pair encoding and dual system groups, which we will discuss in Appendix A. Because this particular scheme with constant-size ciphertexts is only defined in terms of pair encoding and dual system groups, we will show how the generic construction that is given in Appendix A can be applied such that an ABE scheme that fits the description of ABE in Definition 16 in Appendix B. In Appendix C, we decentralize the setup of the resulting scheme in an attempt to make it more suitable for DECODE. Finally, we conclude our work in Chapter 5 by evaluating our results.

Chapter 2

Preliminaries

Before we introduce attribute-based encryption, we will need to establish some mathematical background, along with certain notations first.

2.1 Notations

If an element is chosen uniformly at random from some finite set S, then we denote this as $x \in_R S$. We will denote the order of g in $\mathbb{G}$ as $\mathrm{ord}(g)$, which is the unique number a such that $g^a \equiv 1_{\mathbb{G}}$, and for all $0 < b < a$ we have $g^b \not\equiv 1_{\mathbb{G}}$. We define $\mathcal{H} : \{0,1\}^* \to \mathbb{G}$ as a hash function, which is preimage, second preimage, and collision resistant, and can be mapped to any group $\mathbb{G}$ or set S. If we only use one or two of those properties, we will mention this[1]. We will use $y \leftarrow \mathcal{H}(x)$ to indicate an assignment of a value to a particular integer y (in this case we have used the hash function, but this may be any function, algorithm or other mathematical operation). If some function or algorithm yields no output, then we use $\perp$ to indicate this. We will also use a shorter notation for the set of integers $\{1, 2, ..., n\}$, namely $[1, n]$. When $\mathbb{Z}_p$ or $\mathbb{Z}_N$ are used without further explanation, then p is usually used for prime values of p, and N for general values (or composite numbers if this is indicated). We will often use shorthand for summations, i.e. $\sum_i x_i$ instead of $\sum_{i \in S} x_i$, but only if it is clear over which set is summed. Furthermore, we distinguish elements from vectors and matrices by using boldfaced characters, e.g. $\mathbf{s}$ for a vector and $\mathbf{A}$ for a matrix. We denote $\mathbf{A}_{ij}$ to indicate the entry in the i-th row and the j-th column, and $\mathbf{A}^\intercal$ to be the transpose of the matrix $\mathbf{A}$.

2.2 Statistical distance and indistinguishability

One of the notions that will play an important role in this work is the notion of statistical distance. This is a measure that computes the indistinguishability between two probability distributions.

Definition 1 (Statistical distance) *Let X and Y be two random variables, and V be the set of all possible values of both X and Y. The statistical distance between X and Y is defined as*

$$\Delta(X; Y) = \frac{1}{2} \sum_{v \in V} |\Pr[X = v] - \Pr[Y = v]|.$$

The value of the statistical distance is related to the term ‘indistinguishability’ which means that we cannot distinguish two probability distributions by observing

[1] Sometimes we only need one of the properties in a protocol in order to ensure security, and then we can avoid having to resort to the random oracle model, which we will discuss later.

the values. There are three types of indistinguishability, namely perfect, statistical and computational indistinguishability.

Definition 2 (Negligible functions) *A function $f : \mathbb{N} \to \mathbb{R}$ is called negligible if for every $\delta \in \mathbb{N}$, there is some $x_0 \in \mathbb{N}$ such that for all $x \geq x_0$ we have $f(x) \leq 1/x^\delta$.*

We use this to define the notion of indistinguishability:

Definition 3 (Indistinguishability) *Let $X = \{X_i\}_{i \in I}$ and $Y = \{Y_i\}_{i \in I}$ be two infinite families of random variables X_i and Y_i over some set of possible values V_i, such that X_i, Y_i and V_i somehow depend on index i, where I denotes the set of possible values of i. For each $i \in I$, we consider the statistical distance $\Delta(X_i, Y_i)$:*

- *If $\Delta(X_i; Y_i) = 0$ for all i, then X and Y are perfectly indistinguishable. We denote this as $X \equiv Y$.*
- *If $\Delta(X_i; Y_i)$ is negligible in a function of $|i|$ (the bit length of i), then X and Y are statistically indistinguishable, which we denote as $X \cong Y$.*
- *If for all probabilistic polynomial-time (PPT) algorithms D that decide 0 or 1 for some given random variable X_i or Y_i and possible value $v \in V_i$[2], we have that $|\Pr[D(X_i; v) = 1] - \Pr[D(Y_i; v) = 1]|$, and therefore $\Delta(D(X_i); D(Y_i)) = \frac{1}{2}\sum_{v \in V_i} |\Pr[D(X_i; v) = 1] - \Pr[D(Y_i; v) = 1]|$, is negligible as a function of $|i|$, then X and Y are computationally indistinguishable, which we denote as $X \approx Y$.*

2.3 Secret sharing

Secret sharing is a cryptographic primitive that concerns the art of sharing a secret among different parties. The informal idea of *threshold* secret sharing is that for any threshold t out of n parties that partake in the 'sharing', a group of $t - 1$ or fewer parties cannot reconstruct the secret. This is done by exploiting the nature of some mathematical structures. For instance, in Shamir's secret sharing scheme, Shamir uses the fact that we can only recover the coefficients of a $(t-1)$-degree polynomial if we know at least t different points on that polynomial.

Definition 4 (Secret sharing) *A secret sharing scheme for a dealer $\mathcal{D}$ and participants $\mathcal{P}_1, ..., \mathcal{P}_n$ consists of two algorithms:*

- ***Distribution:*** *Dealer $\mathcal{D}$ shares a secret s in such a fashion that each participant $\mathcal{P}_i$ obtains a share s_i.*
- ***Reconstruction:*** *Any qualified subset of participants $Q \subseteq [1, n]$ can recover the secret by pooling their shares together, i.e. s_i for all $i \in Q$.*

Then Shamir's secret sharing scheme is defined as follows:

Definition 5 (Shamir's (t, n)-secret sharing scheme [Sha79]) *Let $\mathcal{D}$ denote a dealer, and $\mathcal{P}_1, ..., \mathcal{P}_n$ the participants. Let $t \in [1, n]$ be the threshold, i.e. we will need at least t participants in order to retrieve the secret s. This secret s is an element in $\mathbb{Z}_p$, where p denotes a prime larger than n, such that all participants can be uniquely represented in $\mathbb{Z}_p$. Then s is shared as follows:*

[2]That is, $D(X_i; v) = 1$ means that given value $v \in V_i$, D decides that X_i can take on value v.

- ***Distribution:*** *Dealer $\mathcal{D}$ picks a random polynomial of degree $t-1$ over $\mathbb{Z}_p$ with coefficients in $\mathbb{Z}_p$, i.e. $a(X) = \sum_{i=0}^{t-1} a_i X^i$, where $a_0 = s$ and $a_i \in_R \mathbb{Z}_p$. Each participant $\mathcal{P}_i$ gets assigned their own share $s_i = a(i)$.*

- ***Reconstruction:*** *Any group of t participants $Q \subseteq [1, n]$ can recover the secret s by Lagrange interpolation, i.e.*

$$s = \sum_{i \in Q} \lambda_{Q,i} s_i \text{ such that } \lambda_{Q,i} = \prod_{j \in Q \setminus \{i\}} \frac{j}{j-i}.$$

Note that this form of secret sharing is perfect in the sense that if any group of participants $Q \subseteq [1, n]$ such that $|Q| < t$ tries to recover the secret, they will fail to do so, and more importantly, they will not learn any information about the secret s.

Whereas secret sharing in itself is already an interesting primitive, it is also the building block of other cryptographic primitives such as attribute-based encryption, which we will discuss shortly.

We can categorize a secret sharing scheme as linear when they enjoy a certain property: the secret is reconstructed with a linear function of its shares. As we can see, Shamir's SSS is an example of a linear secret sharing scheme, because if s and s' denote two secrets which are shared with Shamir's SSS, and the shares are denoted as s_i and s'_i, then $s_i + s'_i$ denotes the share of secret $s + s'$.

More generally, we give a definition of linear secret sharing schemes:

Definition 6 (Linear Secret Sharing Schemes (LSSS) [Bei96]) *A secret sharing scheme over a set of participants $\mathcal{P}$ is called linear (over $\mathbb{Z}_p$) if*

(i) The shares for each participant form a vector over $\mathbb{Z}_p$.

(ii) There exists an $n_1 \times n_2$ matrix $\mathbf{A}$ called the share-generating matrix. For all $i \in [1, n_1]$, the i-th row of $\mathbf{A}$ is denoted by A_i, and we define ρ to be the function that maps rows to a subset of participants $\mathcal{P}$. Let $\mathbf{v} = (s, v_2, ..., v_{n_2})$ be a column vector, where $s \in \mathbb{Z}_p$ denotes the secret to be shared, and $v_2, ..., v_{n_2} \in_R \mathbb{Z}_p$ are chosen uniformly at random, then $\mathbf{Av}$ is the vector of n_1 shares of the secret s. The i-th element of $\mathbf{Av}$, denoted as $(\mathbf{Av})_i$, is the share that is assigned to participant $\rho(i)$.

Remark: Note that n_1 denotes the number of participants that partake in the sharing, but there is not necessarily a threshold like in Shamir's SSS. That is, we can write each linear threshold secret sharing scheme (e.g. Shamir's SSS) in terms of a share-generating matrix, but not each linear secret sharing scheme that can be written as a share-generating matrix is a threshold scheme. To explain this in more detail, we will use the notion of access structures in Section 2.6.

2.4 Verifiable secret sharing

Secret sharing schemes are a useful tool, but they require the honesty of the participants in the protocol to some extent. As we have shown, an unauthorized group of participants cannot retrieve the secret, which is security aspect that we wish to impose. However, we might also want the participants in the protocol to be honest in the sense that the share that they use to retrieve the secret is actually the share that the participant received in the distribution algorithm, and not some slightly altered version of it.

In verifiable secret sharing (VSS), the dealer also produces some extra information regarding the share of a participant in the open such that other participants can check during the reconstruction step whether the share that the participant provides is actually the same share that was received during the distribution. This extra information is also called the verification key.

An example of a verifiable secret sharing scheme is Feldman's VSSS [Fel87]. Feldman extends Shamir's secret sharing scheme with verification keys such that the shares that the participants provide during the reconstruction step are verifiable. The scheme is defined as follows:

Definition 7 (Feldman's VSSS [Fel87]) *Let $\mathcal{D}$ denote a dealer, and $\mathcal{P}_1, ..., \mathcal{P}_n$ the participants. Let $t \in [1, n]$ be the threshold, i.e. we will need at least t participants in order to retrieve the secret s. This secret s is an element in $\mathbb{Z}_p$, where $p > n$ denotes a prime. We also let $\mathbb{G}$ be a group of prime order p such that g is a generator and the discrete log problem is supposed to be intractable. Then s is shared as follows:*

- ***Distribution:*** *Dealer $\mathcal{D}$ picks a random polynomial of degree $t-1$ with coefficients in $\mathbb{Z}_p$, i.e. $a(X) = \sum_{i=0}^{t-1} a_i X^i$, where $a_0 = s$ and $a_i \in_R \mathbb{Z}_p$. Each participant $\mathcal{P}_i$ gets assigned their own share $s_i = a(i)$. In addition, dealer $\mathcal{D}$ also broadcasts 'commitments' $B_j = g^{a_j}$ for each $j \in [0, t-1]$. Each participant $\mathcal{P}_i$ verifies that*

$$g^{s_i} = \prod_{j=0}^{t-1} B_j^{i^j} \tag{2.1}$$

  holds, such that the participant knows that share s_i was generated correctly.

- ***Reconstruction:*** *Any group of t participants $Q \subseteq [1, n]$ can recover the secret s as in Shamir's SSS in Definition 5. Each participant can verify whether the contributed share s_i of participant $\mathcal{P}_i$ is correct by using Eq. 2.1.*

2.5 Removing the dealer: decentralized VSSS

We can use the notion of verifiable secret sharing in, among other things, distributed key generation algorithms, which is useful in e.g. (t, n)-threshold cryptosystems such as [Ped91]. In such distributed key generation protocols, however, we often want to get rid of the central authority: the dealer. We can do this by letting each participant act as a dealer, and run an instance of Feldman's VSSS. Then each participant can generate a part of the secret, and share this with the other participants. After this, each participant will have n shares. If each participant adds all of his own shares, he will obtain one share of the same secret that was generated by all the participants, i.e. the sum of all secrets that were generated by the participants. The reason that this works is because Shamir's SSS, and therefore also Feldman's VSSS, is linear: adding the shares of two secrets yields one share of the sum of those secrets.

So now we will use this to define decentralized verifiable secret sharing, which is based on the distributed key generation in [Ped91], but works more generally for our work:

Definition 8 (Decentralized VSSS [Ped91]) *For participants $\mathcal{P}_1, ..., \mathcal{P}_n$, let $t \in [1, n]$ be the threshold, and the rest of the parameters as defined in Definition 7*

- ***Distribution:*** *Each participants $\mathcal{P}_i$ picks a random polynomial of degree $t-1$ with coefficients in $\mathbb{Z}_p$, i.e. $a_i(X) = \sum_{k=0}^{t-1} a_{i,k} X^k$, where $a_{i,0} = s_i$ denotes the* *secret to be shared with the rest of the participants by $\mathcal{P}_i$ and $a_{i,k} \in_R \mathbb{Z}_p$. Each participant $\mathcal{P}_j$ will receive their own share $s_{i,j} = a_i(j)$ from participant $\mathcal{P}_i$. In addition, $\mathcal{P}_i$ also broadcasts 'commitments' $B_{i,k} = g^{a_{i,k}}$ for each $k \in [0, t-1]$. Each participant $\mathcal{P}_j$ verifies that*

$$g^{s_{i,j}} = \prod_{k=0}^{t-1} B_{i,k}^{j^k} \tag{2.2}$$

holds, such that participant $\mathcal{P}_j$ knows that share $s_{i,j}$ was generated by $\mathcal{P}_i$ correctly. Each participant will add the shares, i.e. $\bar{s}_j = \sum_{i=1}^{n} s_{i,j}$, such that $\bar{s}_j$ denotes the share of secret $\bar{s} = \sum_{i=1}^{n} s_i$.

- ***Reconstruction:*** *Any group of t participants $Q \subseteq [1, n]$ can recover the secret $\bar{s}$ as in Shamir's SSS in Definition 5. Each participant can verify whether the contributed secret share $\bar{s}_j$ of participant $\mathcal{P}_j$ is correct by checking*

$$g^{\bar{s}_j} = \prod_{i=1}^{n} \left(\prod_{k=0}^{t-1} B_{i,k}^{j^k} \right). \tag{2.3}$$

2.6 Access structures

Another important part in attribute-based encryption is the notion of access structures. There are several types of access structures that are used in ABE schemes. The main reason for this is that in the earlier proposals of ABE schemes, more simplistic access structures were used in the construction of the schemes, whereas more research finally resulted in the use of more expressive access structures that would allow any access policy to be imposed on the encrypted data, as long as the possession of a certain attribute does not have a negative influence on the access policy, i.e. having a certain credential may not result in a denial of access. This means that the access structure is monotone.

Whereas monotone access structures do not allow any negations in the access policies, non-monotone structures do, which initially sounds like a less restrictive definition of access structures, but will prove itself to be, ironically so, a little more restrictive in a sense when it comes to implementations of ABE schemes. The intuitive idea behind this is that it is evidently easier to prove possession of an attribute than the lack thereof. For instance, proving possession might just entail the revelation of said attribute, whereas proving absence of that attribute might be much more difficult.

First, we give a formal definition of (monotone) access structures:

Definition 9 ((Monotone) Access Structures) *Let $\{\mathcal{P}_1, ..., \mathcal{P}_n\}$ be a set of participants. A collection $\mathbb{A} \subseteq 2^{\{\mathcal{P}_1,...,\mathcal{P}_n\}}$ is monotone if for all B, C holds: $B \in \mathbb{A}$ and $B \subseteq C$, then also $C \in \mathbb{A}$. An (monotone) access structure is a (monotone) collection $\mathbb{A}$ of non-empty subsets of $\{\mathcal{P}_1, ..., \mathcal{P}_n\}$. The sets in $\mathbb{A}$ are called the authorized sets, and the sets not in $\mathbb{A}$ are called the unauthorized sets.*

Notation: In order to distinguish 'normal' sets from sets of attributes, we use $\mathcal{S}$. If a set of attributes $\mathcal{S}$ satisfies an access structure $\mathbb{A}$, we denote this as $\mathbb{A} \models \mathcal{S}$.

It is clear that an access structure is non-monotone if it does not satisfy the property that for all B, C holds that if $B \in \mathbb{A}$ and $B \subseteq C$, then also $C \in \mathbb{A}$. Informally, this means that the presence of an attribute might result in an unauthorized set, whereas the absence of that attribute would still be an authorized set. This would happen, for instance, if an access policy contains a negation: $\neg\text{Dutch} \wedge \text{Student}$, i.e. we address international students. Having the set of attributes $\{\text{Student}\}$ would result in access, whereas the extension of this set, $\{\text{Student}, \text{Dutch}\}$, would result in a denial of access.

This also implicitly sketches a part of the problem: just because a user has a set of attributes $\{\text{Student}\}$, it does not mean that said user is not Dutch. It might simply mean that the user in question does not have that attribute. So in order to solve this problem, we might have to require each user to either have the attribute, or the negation of the attribute. How this affects ABE schemes will be shown later, as it might force us to make sacrifices in the structure of our ABE scheme, which might also have a negative influence on the scalability and other properties. As it will turn out, for instance, the number of attributes in the system is bounded and fixed after the setup (in Section 3.3 we will see why this is restrictive). Despite this, we might still want to allow for non-monotone access structures, in which case there are attribute-based encryption systems with non-monotone access structures (e.g. [OSW07]), but they cannot have certain properties that we might wish to impose on the ABE system that we are trying to construct.

Aside of the monotonicity of access structures, there is also the question of how we express an access policy, and how expressive they may be. For instance, there are ABE schemes that only allow conjunctions ($\wedge$) [CN07; Yu+10], and only on the attribute or its negation, so the access policies require users to have all of the attributes (or their negation). There are also ABE schemes that allow the attributes to be multivalued, making it a bit more expressive than just allowing the presence or absence of an attribute [NYO08; Emu+09]. Other schemes use a threshold, which requires a user to have at least t out of n listed attributes, but do not allow negations in the access policies [SW05].

Others make use of access trees to represent monotone access policies [BSW07; Cha07], which allow conjunctions and disjunctions. These access trees apply Shamir's secret sharing scheme in a fashion that we will explain in Section 4.1.

There is another way to make use of such expressive, monotone access policies, namely by using the aforementioned linear secret sharing schemes (see Definition 6) in the representation of the access structures [Wat11; LW11].

2.6.1 LSSS matrices as access structures

In [Bei96], Beimel shows that there is a close relationship between linear secret sharing schemes (LSSS) and a linear algebraic model of computation called span programs, which we can use to express access structures as matrices, and is used in a wide variety of schemes (e.g. [Wat11; LW11]).

As we had shown in Definition 6, we can write any linear secret sharing scheme in terms of a share-generating matrix $\mathbf{A}$ and some function ρ that maps the rows to participants in $\mathcal{P}$. More generally, we have that for any set of participants $\mathcal{P}$, we can define a collection of subsets of $\mathcal{P}$ such that it consists of the authorized subsets and a collection of unauthorized subsets as in Definition 9. In other words, a share-generating matrix $\mathbf{A}$ represents an access structure, and conversely, any monotone access structure can be represented with a share-generating matrix.

Hence, we can write any monotone access policy in terms of an LSSS matrix $\mathbf{A}$ and function ρ, i.e. $\mathbb{A} = (\mathbf{A}, \rho)$. Then we can distribute secret s by generating $v_2, ..., v_{n_2} \in_R \mathbb{Z}_p^{n_2}$ and assigning $A_i v$ to participant $\rho(i)$. Then Beimel shows that any secret sharing scheme that satisfies Definition 6 enjoys the linear reconstruction property: suppose that we have an LSSS matrix for the access structure $\mathbb{A}$, i.e. $\mathbb{A} = (\mathbf{A}, \rho)$. Let $\mathcal{S}$ be any authorized set, i.e. $\mathbb{A} \models \mathcal{S}$, and let $\Upsilon \subseteq [1, n]$ be defined as $\Upsilon = \{i \in [1, n] : \rho(i) \in \mathcal{S}\}$. Let $A_i v$ be the share of participant $\rho(i)$, and $\omega_i \in \mathbb{Z}$ for all $i \in \Upsilon$ such that $\sum_{i \in \Upsilon} \omega_i A_i = (1, 0, ..., 0)$. Then $\sum_{i \in \Upsilon} \omega_i A_i v = s$.

We can generate such matrices $\mathbf{A}$ by applying algorithms such as the ones as devised in [LW10], which converts any access policy (in terms of a Boolean formula) into such matrix.

2.7 Zero-knowledge proofs of knowledge

Another cryptographic primitive that we will be using is the notion of zero-knowledge proofs. These are identification protocols in which the prover shows to a verifier that he knows a certain secret without having to reveal that secret or any information regarding it in plain view.

More specifically, we will be using the notion of Σ-protocols, which are identification protocols that satisfy certain requirements that we will discuss later. In these protocols, a prover wants to prove knowledge of a value to a verifier, for instance the knowledge of a secret key corresponding to a public key. The prover does this by committing to a random value, a nonce, which we will call an *announcement*[3]. Then the verifier generates a random value, which is called the *challenge*, and sends it to the prover, who has to use it in the last phase of the protocol. In this last phase, the prover generates a response, and sends it to the challenger. Then the announcement, challenge and response can be used to verify the knowledge of the value.

Before we get to the definition of Σ-protocols, we will first define the notion of a binary relation, $R = \{(v, w)\} \subseteq V \times W$, which is the set of tuples in which $v \in V$ denotes the common variable of the prover and verifier, which is typically the 'public key' or other public variable that the prover wants to use to prove the knowledge of some secret, and $w \in W$ denotes the witness, which is typically the secret in question.

We will denote the triple $(a; c; r)$ as the *conversation* that was generated by a protocol, in which a denotes the announcement, c denotes the challenge and r denotes the response, and it will prove the knowledge of witness w for some given public v. The conversation is *accepting* if $(a; c; r)$ is well-defined and satisfies the verification check.

Definition 10 (Σ-protocols [Cra96; Sch17]) *A protocol between a prover and verifier is a Σ-protocol for relation R if it satisfies the following properties:*

- ***Completeness:*** *If the prover and verifier follow the protocol, then the verifier accepts.*
- ***Special soundness:*** *For any given $v \in V$, and any pair of accepting conversations $(a; c; r)$ and $(a; c'; r')$ on this v such that $c \neq c'$, we can extract a witness $w \in W$ in polynomial time such that $(v, w) \in R$.*
- ***Special honest-verifier zero-knowledgeness:*** *For some given $v \in V$ (for which $w \in W$ such that $(v, w) \in R$ exists) and challenge c, the distributions of the simulated conversations $(a; c; r)$ and the conversations that are generated by running the protocol between an honest prover (that knows w) and verifier are perfectly indistinguishable. Furthermore, some simulator is also required to*

[3]This term was coined by Schoenmakers in [Sch17] and is a contraction of the words 'commitment' and 'nonce'.

$$\begin{array}{ccc}
\text{Prover} & & \text{Verifier} \\
(\text{Knows}: x) & & (A = g^x) \\
u \in_R \mathbb{Z}_p & & \\
a \leftarrow g^u & \xrightarrow{\quad a \quad} & \\
 & & c \in_R \mathbb{Z}_p \\
 & \xleftarrow{\quad c \quad} & \\
r \leftarrow u + cx & \xrightarrow{\quad r \quad} & g^r \overset{?}{=} aA^c
\end{array}$$

FIGURE 2.1: Schnorr's protocol

$$\begin{array}{ccc}
\text{Prover} & & \text{Verifier} \\
(\text{Knows}: x, y) & & (B = g^x h^y) \\
u_1, u_2 \in_R \mathbb{Z}_p & & \\
a \leftarrow g^{u_1} h^{u_2} & \xrightarrow{\quad a \quad} & \\
 & & c \in_R \mathbb{Z}_p \\
 & \xleftarrow{\quad c \quad} & \\
r_1 \leftarrow u_1 + cx & & \\
r_2 \leftarrow u_2 + cy & \xrightarrow{\quad r_1, r_2 \quad} & g^{r_1} h^{r_2} \overset{?}{=} aB^c
\end{array}$$

FIGURE 2.2: Okamoto's protocol

produce arbitrary accepting conversations for any v for which no witness exists, for any given challenge c.

In this work, we will need two different Σ-protocols, namely Schnorr's identification protocol [Sch91] and Okamoto's identification protocol [Oka92]. We will also need EQ-compositions of the protocols in order to prove 'correctness' of certain computations. For instance, if we want to some party to compute A and B such that $A = g^x$ and $B = g^y h^x$, he could prove with a Σ-protocol that he has indeed done this in the correct way.

2.7.1 Schnorr's identification protocol

In Fig. 2.1, the schematic overview of Schnorr's identification protocol is depicted, in which the prover proves knowledge of x in $A = g^x$, for some generator g and $x \in \mathbb{Z}_p$, where A is known to both the prover and verifier. It is easily proven that Schnorr's identification protocol as depicted in Fig. 2.1 is a Σ-protocol, though we have omitted the proof in this work.

2.7.2 Okamoto's identification protocol

Okamoto's identification protocol, as depicted in Fig. 2.2, is very similar to Schnorr's identification protocol, but instead the prover proves knowledge of two values in $B = g^x h^y$, i.e. x and y. In order for it to be a Σ-protocol, we have to assume that the $\log_g h$ is unknown to anyone, otherwise the prover can forge multiple witnesses, given the public input B. It is also proven that the protocol as depicted in Fig. 2.2 is a Σ-protocol.

2.7.3 EQ-composition

In order to show that two public values have a common witness, we can use an EQ-composition of two Σ-protocols. That is, if we have a Σ-protocol for relation R, then we can also create a Σ-protocol for relation $\{(v_1, v_2; w) : (v_1, w) \in R \wedge (v_2; w) \in R\}$. The basic idea of such a composition is that we use the same challenge and the same response for the part that we want to prove equal. To give a better idea of how this works, we give a Σ-protocol that proves knowledge of x and y in $A = g^x \wedge B = g^y h^x$. As we can see in Fig. 2.3, we will need instances of Schnorr's and Okamoto's protocols, and then we apply the EQ-composition on the both of them.

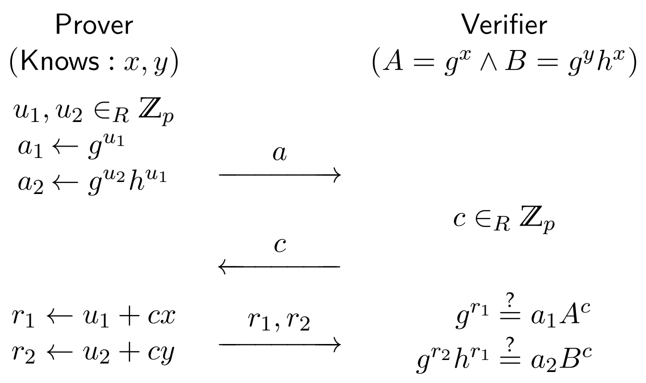

Prover
(Knows : x, y)
Verifier
$(A = g^x \wedge B = g^y h^x)$
$u_1, u_2 \in_R \mathbb{Z}_p$
$a_1 \leftarrow g^{u_1}$
$a_2 \leftarrow g^{u_2} h^{u_1}$
a
$c \in_R \mathbb{Z}_p$
c
$r_1 \leftarrow u_1 + cx$
$r_2 \leftarrow u_2 + cy$
r_1, r_2
$g^{r_1} \stackrel{?}{=} a_1 A^c$
$g^{r_2} h^{r_1} \stackrel{?}{=} a_2 B^c$

FIGURE 2.3: EQ-composition of Schnorr's and Okamoto's protocols

2.7.4 Non-interactive zero-knowledge proofs

In some setups, the interactivity between the prover and verifier is not desired or even possible. We can imagine that sometimes, a prover wants to prove the knowledge of a value, but not to one particular verifier. In this case, a non-interactive zero-knowledge proof would be desirable. As Fiat and Shamir have shown in [FS86], it is fairly easy to convert an interactive Σ-protocol into a non-interactive Σ-proof, a method that is called the Fiat-Shamir heuristic. To do this, all we need is a hash function $\mathcal{H} : \{0,1\}^* \to \{0,1\}^k$ for some properly chosen bit length k.

Suppose that the challenge is defined as $c \leftarrow \mathcal{H}(a, v) \bmod p$, where a denotes an announcement, and $v \in V$ denotes the public value. Then r is the response to the challenge c. The verification equation always takes a into account. So we can easily rewrite it as a function of v, r and c, i.e. $a = \varphi(v; c; r)$. This will be our input to the hash function: so we have to check whether $c \stackrel{?}{=} \mathcal{H}(\varphi(v; c; r), v)$ holds, and if so, then the prover has shown that he knows witness w such that $(v, w) \in R$.

Definition 11 (Σ-proof [FS86; Sch17]) *Let $\mathcal{H}$ be a cryptographic hash function. For any Σ-protocol, a (non-interactive) Σ-proof for relation R is defined in terms of two algorithms:*

- ***Proof generation:*** *Given $(v, w) \in R$, a Σ-proof is a challenge-response pair $(c; r)$, such that for announcement a, we have $c \leftarrow \mathcal{H}(a, v) \bmod p$, and response r was generated in the same fashion as in the Σ-protocol.*

- ***Proof verification:*** *For $v \in V$, Σ-proof $(c; r)$ is accepted if and only if $c \stackrel{?}{=} \mathcal{H}(\varphi(v; c; r), v)$ holds. Here, $\varphi(v; c; r)$ is the function that we have obtained after rewriting the verification equation in the Σ-protocol.*

So from the Σ-protocol in Fig. 2.1 we can make a non-interactive Σ-protocol:

Example 12 (Schnorr's Σ-proof) *Let $\mathcal{H}$ be a cryptographic hash function.*

- ***Proof generation:*** *Let $A = g^x$, and pick $u \in_R \mathbb{Z}_p$ such that $a \leftarrow g^u$, as in Fig. 2.1. Let the challenge be $c \leftarrow \mathcal{H}(a, A) \bmod p$, and the response $r \leftarrow u + cx \bmod p$. Then the output is the Σ-proof $(c; r)$.*

- ***Proof verification:*** *For any given A, a Σ-proof $(c; r)$ is considered valid if and only if $c \stackrel{?}{=} \mathcal{H}(g^r / A^c, A)$ holds.*

2.8 Group homomorphisms

We can define functions from groups $\mathbb{G}$ to other groups $\mathbb{H}$. A special type of function that preserves the structure is called a homomorphism. If $\cdot$ is an operation on $\mathbb{G}$, and $\odot$ is an operation on $\mathbb{H}$, and we define function $\varphi : \mathbb{G} \to \mathbb{H}$ such that for all $g_1, g_2 \in \mathbb{G}$ holds that $\varphi(g_1 \cdot g_2) = \varphi(g_1) \odot \varphi(g_2)$, then φ is called a homomorphism.

If φ is a homomorphism, then we can define two sets based on this function. The first is the kernel of φ, which is the set of all elements in $\mathbb{G}$ that are mapped to the identity in $\mathbb{H}$, i.e. $\text{Kernel}(\varphi) := \{g \in \mathbb{G} : \varphi(g) = 1_{\mathbb{H}}\}$. The second is the image of φ, which is the set of elements h in $\mathbb{H}$ such that there is some $g \in \mathbb{G}$ such that $\varphi(g) = h$, i.e. $\text{Image}(\varphi) := \{\varphi(g) : g \in \mathbb{G}\} = \varphi(\mathbb{G})$.

2.9 Bilinear maps

In a somewhat similar fashion, we can define a map e on two possibly different groups $\mathbb{G}$ and $\mathbb{H}$ of prime order p, i.e. $e : \mathbb{G} \times \mathbb{H} \to \mathbb{G}_T$, where $\mathbb{G}_T$ is called the target group, such that the following properties hold:

(i) *(Bilinearity I)* For all $a, b \in \mathbb{Z}_p$ we have $e(g^a, h^b) = e(g, h)^{ab}$.

(ii) *(Bilinearity II)* For all $g_1, g_2 \in \mathbb{G}$ and $h_1, h_2 \in \mathbb{H}$ we have $e(g_1g_2, h_1h_2) = e(g_1, h_1)e(g_1, h_2)e(g_2, h_1)e(g_2, h_2)$.

(iii) *(Non-degeneracy)* If $g \in \mathbb{G}$ and $h \in \mathbb{H}$ are no identity elements in $\mathbb{G}$ and $\mathbb{H}$ respectively, then $e(g, h) \neq 1_{\mathbb{G}_T}$.

Sometimes in literature, a property called 'computability' is included, which simply means that $e(g, h)$ is efficiently computable.

Remark: In some literature, e is not defined over two groups of prime order, but rather over groups of composite order $N = p_1p_2p_3$, where p_i are prime numbers for all i. Let $\mathbb{G}$ and $\mathbb{G}_T$ be two groups of order N, and $e : \mathbb{G} \times \mathbb{G} \to \mathbb{G}_T$ a map such that bilinearity holds, and non-degeneracy is defined slightly differently: i.e. there is some $g \in \mathbb{G}$ such that $e(g, g)$ has order N in $\mathbb{G}_T$.

2.10 Security assumptions

In cryptography, the security of cryptosystems is often reduced to well-established complexity assumptions. For instance, the security of the Diffie-Hellman key exchange [DH76] can be reduced to the decisional Diffie-Hellman assumption, which can be reduced to the complexity of the discrete log problem.

In this work, we will consider a lot of different ABE schemes, of which the security is not always reducible to the same security assumption. However, a multitude of schemes is reducible to the decisional bilinear Diffie-Hellman (DBDH) assumption, or to a slight modification of it. In the second part of this work, we will consider an ABE scheme which security depends on the symmetric external Diffie-Hellman (SXDH) assumption. In both cases, we assume a security parameter λ and group $\mathbb{G}$ of large prime order p (of at least λ bits) with generator g.

For the DBDH assumption, we assume that e is a non-degenerate bilinear map such that $e : \mathbb{G} \times \mathbb{G} \to \mathbb{G}_T$.

Assumption 13 (Decisional bilinear Diffie-Hellman assumption [SW05]) *Suppose that $a, b, c, z \in \mathbb{Z}_p$ are picked at random. Then the assumption is that we cannot distinguish $(A = g^a, B = g^b, C = g^c, Z = e(g,g)^{abc})$ from $(A = g^a, B = g^b, C = g^c, Z = e(g,g)^z)$ in polynomial time and with more than negligible advantage.*

For the SXDH assumption, we also define $\mathbb{H}$ as a group of prime order p with generator h, and e as a non-degenerate bilinear map such that $e : \mathbb{G} \times \mathbb{H} \rightarrow \mathbb{G}_T$.

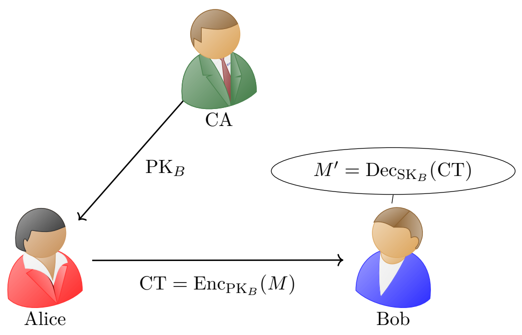
Assumption 14 (Symmetric external Diffie-Hellman assumption [CW14a]) *Suppose that $s_1 \in_R \mathbb{Z}_p$ and $a_1, a_2, s_2 \in_R \mathbb{Z}_p^*$ are picked at random. Then we cannot distinguish $(g^{a_1}, g^{a_2}, g^{a_1 s_1}, g^{a_2 s_1})$ from $(g^{a_1}, g^{a_2}, g^{a_1 s_1}, g^{a_2 s_1 + s_2})$ in polynomial time with more than negligible advantage.*

Analogously, we have that $(h^{a_1}, h^{a_2}, h^{a_1 s_1}, h^{a_2 s_1})$ and $(h^{a_1}, h^{a_2}, h^{a_1 s_1}, h^{a_2 s_1 + s_2})$ are indistinguishable in polynomial time with more than negligible advantage.

Chapter 3

Attribute-Based Encryption

One of the tools that can be used to ensure confidentiality of data, is public-key cryptography. However, most public-key cryptosystems such as RSA [RSA78] and ElGamal [ElG85] have problems with the distribution of keys, as they need to be authenticated in order to be used properly. There are many solutions that attempt to solve this problem, such as the use of certificate authorities (CA), which allows a user to generate a key-pair and then obtain a certificate from a CA that confirms the ownership of that key (see Fig. 3.1 for a schematic overview of this particular setup).


CA
PK_B
$M' = \mathrm{Dec}_{\mathrm{SK}_B}(\mathrm{CT})$
Alice
$\mathrm{CT} = \mathrm{Enc}_{\mathrm{PK}_B}(M)$
Bob
FIGURE 3.1: Public-key encryption (with CA)

Identity-based encryption (IBE) [Sha84] provides us with a way to tie identities to a key-pair such that authentication of keys is trivial. The idea is that one key-pair is generated by some trusted third party, and all other key-pairs are derived from that key-pair in such a fashion that the identities of the users are used in this derivation. The identity could be anything, such as a name or social security number, or any combination of identifiable traits that one wishes to use. The main difference between IBE and regular public-key cryptography is the encryption step: instead of looking up a public key that corresponds to an identity, which is then used the encrypt a message, we can immediately use the identity in order to encrypt the message. In other words, we do not need a trusted third party in the encryption step. Instead, the trusted third party is moved to the key distribution step (see Fig. 3.2 for a schematic overview of IBE).

Now, this system is based entirely on the notion of identity and does not much else than provide us with another way to send messages to other users, however we can also imagine that we do not necessarily want to send anything to specific users, but rather to users that possess certain attributes. For instance, we might want to send a message to everyone that lives in the same street, or to all computer science students, or anything more specific. One way to do this would be to 'simply' look up all identities of these users and then send separate messages to each individual, encrypted with their own respective public key, which provides us with various problems such as finding all of those identities.

Another way to do this would be to apply attribute-based encryption (ABE) [SW05], which was invented by Sahai and Waters and is derived from IBE. Instead of encrypting to separate public keys corresponding to users, the message is encrypted to attributes according to some policy that can be specified by either the key issuer or the encryptor, depending on the 'type' of ABE that we use. In other words, the

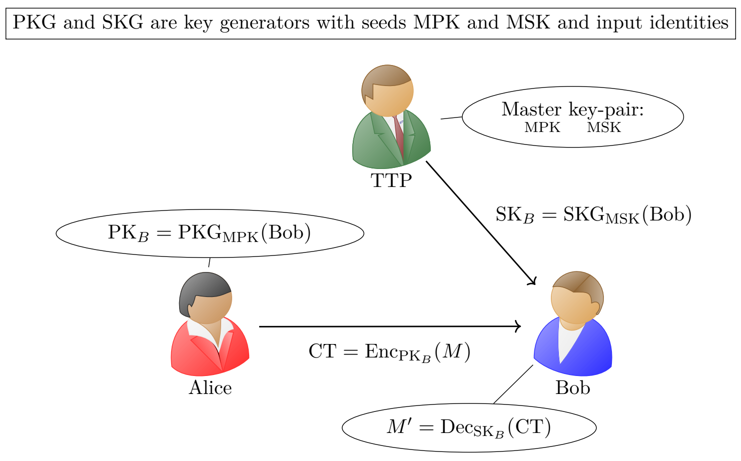

PKG and SKG are key generators with seeds MPK and MSK and input identities
Master key-pair: MPK MSK
TTP
$\mathrm{SK}_B = \mathrm{SKG}_{\mathrm{MSK}}(\mathrm{Bob})$
$\mathrm{PK}_B = \mathrm{PKG}_{\mathrm{MPK}}(\mathrm{Bob})$
$\mathrm{CT} = \mathrm{Enc}_{\mathrm{PK}_B}(M)$
Alice
Bob
$M' = \mathrm{Dec}_{\mathrm{SK}_B}(\mathrm{CT})$

FIGURE 3.2: Identity-based encryption

keys correspond to attributes rather than identities, and the ciphertexts can only be decrypted when the user has the 'right' attributes and their associated keys.

In this scenario, there are different entities. Of course, there are the users in the system, that can both encrypt messages and decrypt ciphertexts. Even outsiders can encrypt messages, as long as the public parameters are known to them. And finally, we need some key issuer or key generation authority (KGA) that generates the master keys, and distributes the secret keys associated with attributes that the user possesses. Moreover, these keys have to be 'personalized' in such a way that we can prevent a so-called collusion attack, which entails that two or more users can collude by combining their keys in such a way that they can decrypt ciphertexts that they cannot decrypt with their own sets of attributes.

Before we give a formal definition of attribute-based encryption, we have to mention that we distinguish between two types of attribute-based encryption, for which the definitions are slightly different. These two types of ABE are called *key-policy* and *ciphertext-policy attribute-based encryption*.

3.1 Key-policy and ciphertext-policy ABE

The idea of attribute-based encryption is that we can define access policies over attributes in the form of Boolean formulas. So for instance, if we want to impose the following policy: the user has to be a mathematics or computer science student from the Netherlands, then we can represent this with the following Boolean formula: $\mathrm{Student} \wedge \mathrm{Dutch} \wedge (\mathrm{Mathematics} \vee \mathrm{ComputerScience})$. Note that this representation is certainly not unique, as we can also write it as $(\mathrm{Student} \wedge \mathrm{Dutch} \wedge \mathrm{Mathematics} \wedge \mathrm{Mathematics}) \vee (\mathrm{Student} \wedge \mathrm{Dutch} \wedge \mathrm{ComputerScience})$ or any other representation that can be rewritten as the same formula.

Now, the question is how we can integrate the policy. First, we convert the access policy (which is represented by a Boolean formula) to some suitable representation of an access structure. Then we can integrate the access structure in the encryption scheme by either imposing the structure on the secret keys or on the ciphertexts. In the first case, the access structure is integrated in the key generation algorithm, and in the second case, the access structure is integrated in the encryption algorithm. Because in the first case, we impose the policy on the secret keys, which is done by the key issuer, this type of ABE is also referred to as *key-policy* attribute-based encryption (KP-ABE), whereas the second type of ABE imposes the policies on the

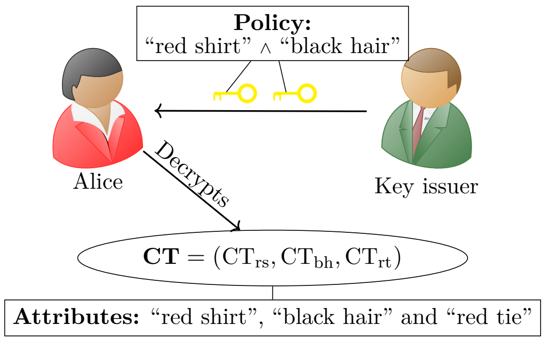

Policy:
"red shirt" ∧ "black hair"
Alice
Key issuer
Decrypts
CT = $(\mathrm{CT}_{\mathrm{rs}}, \mathrm{CT}_{\mathrm{bh}}, \mathrm{CT}_{\mathrm{rt}})$
Attributes: "red shirt", "black hair" and "red tie"

FIGURE 3.4: Example of key-policy attribute-based encryption

ciphertexts, which is done by the encryptor, and is therefore called *ciphertext-policy* attribute-based encryption (CP-ABE) (see Fig. 3.3).

	KP-ABE	**CP-ABE**
Policy imposed on	Keys	Ciphertexts
Policy imposed by	Key issuer	Encryptor

FIGURE 3.3: KP-ABE and CP-ABE and their policies

In practice, the integration of the access structure in either the keys or the ciphertexts (depending on which type of ABE we are dealing with) can be done by exploiting primitives such as secret sharing schemes, which allows us to share a secret value among different attributes. The idea is that the secret can only be recovered if the set of attributes satisfies the access policy. For instance, if our access policy consists of a set of n attributes, for which holds that we need at least $t \in [1, n]$ of those in order to satisfy the policy, then this can clearly be realized by applying Shamir's secret sharing scheme (see Definition 5), which is a (t, n)-threshold scheme. The first ABE scheme that was proposed in [SW05] applies this strategy.

Typically, in KP-ABE, the secret to be shared is some secret value that relates to the master secret key, which is usually some value α in the exponent of one of the public keys that is used to hide the message M in the encryption. This secret value is shared with SSSS, and the shares are each put in the exponent of some cleverly chosen generators. Then, the ciphertexts are related to a set of attributes in some fashion, for which holds that if there is a subset of attributes that consists of at least t attributes that match the set of attributes associated with the secret keys, then the secret keys can be used to decrypt the ciphertext. This works because the ciphertext also contains a secret value s, specified by the encryptor. In [SW05], Sahai and Waters define unique generators for each attribute in the system, such that this specific secret value is only 'added' to the ciphertext for those attributes that the encryptor wishes to use. This can be done by only raising the corresponding generators to the power of the secret value s. In CP-ABE, we can do something similar, but instead use SSSS in the sharing of the secret value in the ciphertext, whereas the keys use the unique generators raised to the power of the master secret key value α.

In Fig. 3.4, an example of such a threshold policy for KP-ABE is depicted, in which the policy specifies that both attributes have to be used in the ciphertext in order to decrypt. Note that, technically, the access policy is expressed as a Boolean formula with a single conjunction, but this is the same as a $(2, 2)$-threshold access structure. This indirectly also sketches the correspondence between Boolean formulas and secret sharing, which we will discuss in more detail in Section 4.1. We also observe in Fig. 3.4 that the attributes associated with the ciphertext satisfy the policy, which

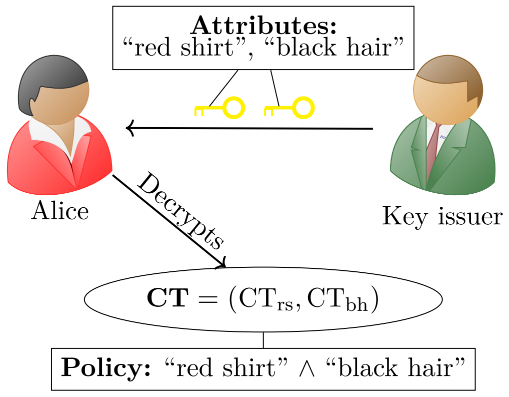

Attributes:
"red shirt", "black hair"
Alice
Key issuer
Decrypts
$\mathbf{CT} = (\mathrm{CT}_{\mathrm{rs}}, \mathrm{CT}_{\mathrm{bh}})$
Policy: "red shirt" $\wedge$ "black hair"

FIGURE 3.5: Example of ciphertext-policy attribute-based encryption

means that Alice can decrypt the ciphertext. Note that the policy and attributes are both attached to the key and ciphertext respectively, which is necessary to know how to combine the keys and ciphertext in order to decrypt. It is also possible to use hidden policies, but we will discuss this in Section 3.9. In Fig. 3.5, we see a similar threshold policy for CP-ABE is depicted, but observe that in this case, the policy is imposed on the ciphertext.

Because we are looking for fine-grained access control of data, controlled by users, i.e. users get to decide who reads data pertaining to or generated by them, we will mostly focus on ciphertext-policy attribute-based encryption in the remainder of this work, as it allows encryptors to specify the access policy, contrary to KP-ABE, which only allows the key issuer to do this.

3.1.1 Formal definitions

In [SW05], Sahai and Waters introduced the notion of attribute-based encryption. However, the first definition was formulated in [Goy+06], which gives a formal definition of key-policy attribute-based encryption.

Definition 15 (KP-ABE [Goy+06]) *A key-policy attribute-based encryption scheme with some key generation authority (KGA), users and a universe of attributes $\mathcal{U}$ (which denotes the set of all of the attributes that are defined in the system) consists of four algorithms:*

(i) ***Setup:*** *This is a randomized algorithm executed by the key generation authority that takes no input other than the security parameter λ. It outputs the public parameters* MPK *and master secret key* MSK*, which is only known to the KGA.*

(ii) ***Encrypt:*** *This is a randomized algorithm executed by any user in the system that takes as input a message M, a set of attributes $\mathcal{S} \subseteq \mathcal{U}$ and public parameters* MPK*. It outputs ciphertext* CT*.*

(iii) ***KeyGen:*** *This is a randomized algorithm executed by the KGA upon a user's request that takes as input an access structure $\mathbb{A}$ (that the user satisfies), the master secret key* MSK *and public parameters* MPK*. It outputs a decryption key* SK*.*

(iv) ***Decrypt:*** *This algorithm takes as input the ciphertext* CT *that was encrypted under attributes $\mathcal{S}$, the user's private key* SK *for access structure $\mathbb{A}$, and the public parameters* MPK*. It outputs message M if $\mathbb{A} \models \mathcal{S}$, and $\perp$ if $\mathbb{A} \not\models \mathcal{S}$.*

Note that whether the decryption algorithm outputs the message or not, depending on whether the set of attributes satisfies the access structure, is also referred to as the correctness property.

As we can see, the encryption algorithm takes a set of attributes as input, whereas the key generation algorithm takes an access structure as input. We will see that this is the other way around for ciphertext-policy attribute-based encryption. The first formal definition of CP-ABE was given in [BSW07], which closely resembles the definition of KP-ABE.

Definition 16 (CP-ABE [BSW07]) *A ciphertext-policy attribute-based encryption scheme with some key generation authority (KGA), users and a universe of attributes $\mathcal{U}$ consists of four algorithms:*

(i) ***Setup:*** *This is a randomized algorithm executed by the key generation authority that takes no input other than the security parameter λ. It outputs the public parameters* MPK *and master secret key* MSK*, which is only known to the KGA.*

(ii) ***Encrypt:*** *This is a randomized algorithm that takes as input a message M, an access structure $\mathbb{A}$ specified by the encryptor and public parameters* MPK*. It outputs ciphertext* CT *such that only a user that possesses a set of attributes $\mathcal{S} \subseteq \mathcal{U}$ that satisfies the access structure can decrypt the message.*

(iii) ***KeyGen:*** *Upon some user's request, this randomized algorithm takes as input a set of attributes $\mathcal{S}$, the master secret key* MSK *and public parameters* MPK *and lets the KGA compute private key* SK *for the user with attributes $\mathcal{S}$.*

(iv) ***Decrypt:*** *This algorithm takes as input the ciphertext* CT *that was encrypted under access structure $\mathbb{A}$, the user's decryption key* SK *associated with the set of attributes $\mathcal{S}$, and the public parameters* MPK*. It outputs message M if $\mathbb{A} \models \mathcal{S}$. If $\mathbb{A} \not\models \mathcal{S}$, it outputs $\perp$.*

Note that whether the decryption algorithm outputs the message or not, depending on whether the set of attributes satisfies the access structure, is also referred to as the correctness property.

3.2 Collusion resistance

As we already briefly mentioned, we do not allow users to collaborate in order to decrypt messages that they are not supposed to decrypt. In other words, if we have two users with sets of attributes $\mathcal{S}_1$ and $\mathcal{S}_2$, and access structure $\mathbb{A}$ such that $\mathbb{A} \models \mathcal{S}_1 \cup \mathcal{S}_2$ holds but $\mathbb{A} \not\models \mathcal{S}_1$ and $\mathbb{A} \not\models \mathcal{S}_2$, then the users should not be able to decrypt the message.

To achieve collusion resistance, the key generation algorithm often generates some random number that 'links' the secret keys together, so they can only be used together. When two users will attempt to pool their secret keys, they will not 'fit' together because the random numbers do not correspond. Alternatively, the secret keys are linked to the user's identity in some fashion. In that case, we can also define collusion resistance for decentralized systems that do not use one central authority to distribute the keys, which we will discuss in Section 3.5.

3.3 Small and large universes

In ABE schemes, the attributes are often represented by integers or group elements, either because of notation purposes, or because the attributes are somehow part of the algorithms. The set of possible attributes that a user can possess is called a universe of attributes, often denoted as $\mathcal{U}$. We distinguish between two types of universes, namely small and large universes. The main difference is that in the former, we are required to choose the universe of attributes in the setup, which means that once we have chosen a set of possible attributes, this set is fixed after the setup, and we cannot add new attributes. Alternatively, during setup, the key issuer sets a 'upper bound' to the number of attributes that are 'eventually' needed. But this bound is required to be polynomial in the security parameter, and restricts the system considerably, especially if the number of public keys grows in the size of the universe of attributes.

Large universe systems, however, do not have this 'upper bound'. They can represent the universe of attributes with all of the elements in $\mathbb{Z}_N$ or some other group of elements of order N, where N is a number of at least λ bits. Recall that λ denotes the security parameter. In this case, the universe of attributes is exponentially large in the security parameter, as opposed to small universes of attributes. A large universe of attributes is therefore practically unbounded. In [SW05], a large universe of attributes is constructed by mapping attribute strings to elements of $\mathbb{Z}_N$ with a collision-resistant hash function $\mathcal{H}$, which is a practical method to represent attributes in the cryptosystem and therefore often used in literature. Also note that contrary to small universes, large universes do not allow the public keys to grow in the size of the universe of attributes, which makes large universe constructions to some extent more scalable than small universe constructions.

	Small	**Large**
$\lvert\mathcal{U}\rvert$	Polynomial in λ	Exponential in λ
$\lvert\mathrm{PK}\rvert$	May be linear in $\lvert\mathcal{U}\rvert$	Independent on $\lvert\mathcal{U}\rvert$

FIGURE 3.6: Small versus large universes

Because clearly, large universe constructions provide us with a way to constantly add new attributes, we prefer large universe constructions in the DECODE setting. Moreover, the size of the public keys is independent on the number of attributes in the system, which also makes the scheme more scalable. However, this also means that non-monotone access structures are out of the question, as to the best of our knowledge, there are no CP-ABE schemes with a large universe of attributes and non-monotone access structures. The reason for this is probably because for each attribute in the attribute universe, we need to specify whether a user possesses it or not. If our scheme were a large universe construction, then we could not realize this: each user would have to indicate of each possible attribute, which is a number that is exponential in the security parameter, whether he has it or not, and the key issuer would have to issue keys for each element in e.g. $\mathbb{Z}_N$ to each user in the system, which is not feasible.

From a functional point of view, we would definitely prefer non-monotone access structures, as it allows us to use more expressive access structures. But from the practical point of view, we also argued in Section 2.6 that we preferred monotone access structures over non-monotone access structures in the implementation, as it is simply easier to prove possession of something than the lack thereof, especially in a large practical setting such as DECODE. Hence, in this case, there is not much of a trade-off.

3.4 Delegatability

In some attribute-based encryption schemes, there is an additional algorithm for delegation of keys. This algorithm allows users that already have secret keys to delegate their keys to other users, which sounds rather trivial, but the idea is that the policy or the set of attributes may be more restrictive than the original. So, for key-policy ABE, the delegation algorithm takes the secret keys associated with some access policy $\mathbb{A}$ as input, and outputs the secret keys associated with a more restrictive policy $\mathbb{A}'$ such that for all sets of attributes $\mathcal{S}$ holds: if $\mathbb{A}' \models \mathcal{S}$, then $\mathbb{A} \models \mathcal{S}$. For ciphertext-policy ABE, the delegation algorithm takes the secret keys associated with some set of attributes $\mathcal{S}$ as input, and outputs the secret keys associated with a smaller set of attributes $\mathcal{S}' \subseteq \mathcal{S}$.

The ability to define such a delegation algorithm in an ABE scheme, which we will refer to as *delegatability*, provides us with other benefits, which we will see in Section 3.10.4. However, because DECODE aims to implement attribute-based encryption in combination with attribute-based credentials, and requires a user in the system to authenticate towards the key generation authority in order to prove that he has the right to possess the requested secret key associated with some attribute, we do not wish to implement such delegation algorithms in the system, as it means that users can share their keys with other users. Moreover, as we will see in the next section, in the multi-authority setting, the secret keys are tied to global identifiers and per definition we require that such secret keys cannot be transformed in secret keys for other global identifiers. Hence, delegatability and decentralization are, to some extent, incompatible with one another.

3.5 (Decentralized) multi-authority ABE

Up until now, we have considered ABE schemes in which we assume a central authority that acts as a trusted third party. However, we are trying to move away from trusting one party, and decentralize the trust that has to be put into the authorities (see Fig. 3.7 for a schematic overview). We are trying to put the DE in DECODE, so to speak.

In the last decades, the decentralization of trust has become more popular in security areas. For instance, blockchains provide a decentralized 'alternative' to centralized banks, and secure multiparty computation makes it possible to distribute the computation of a function in a secure, distributed fashion.

Hence, the decentralization of attribute-based encryption is a logical next step. In 2007, Chase [Cha07] published the first multi-authority (KP-)ABE scheme that – to some extent – distributes the secret key generation among different authorities: each authority will be responsible for their own unique set of attributes, and distributes keys to users that are entitled to those keys. However, this scheme still needs a central authority that distributes a part of the secret keys, so in a sense, we still have one centralized point of trust. Moreover, the CA is able to decrypt all messages in the system, which means that corruption of the CA leads to a massive privacy violation. But despite its disadvantages, Chase's work paves the way for fully decentralized ABE, and she also introduces the notion of global identifiers, which ties each user to an identity in the system. This way, authorities can use the global identifier to 'tie the keys together' in such a way that collusion between users can be avoided. This notion will prove itself paramount in the design of other decentralized ABE schemes.

In [CC09], Chase and Chow showed how we can remove the central authority by letting each authority contribute to the generation of the master secret key. Then the

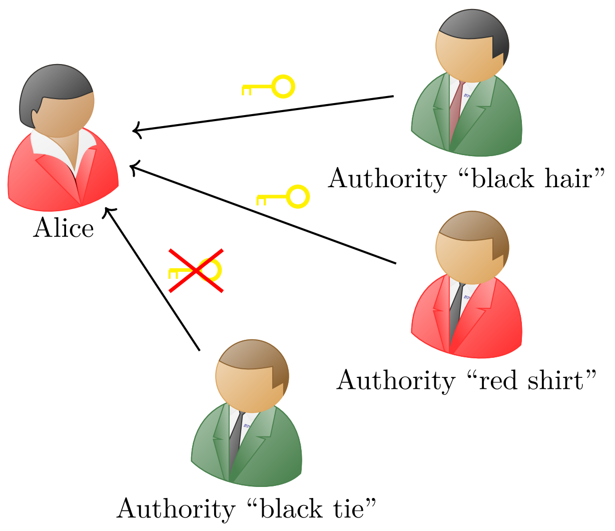


FIGURE 3.7: Decentralized (multi-authority) CP-ABE

master secret key is divided in different pieces for each user by using a pseudo-random function that is dependent on the global identifier of the user. This ensures that each user gets a different ‘sharing’ of the master secret key (and therefore avoids collusion) while still allowing the sharing to sum to the master secret key.

Whereas the aforementioned multi-authority schemes are both key-policy based, there are also decentralized multi-authority ciphertext-policy ABE schemes. In 2011, Lewko and Waters [LW11] published a decentralized CP-ABE scheme (see Fig. 3.7 for a schematic overview) that only uses a ‘central authority’ to set up the system parameters, such as the groups in which the system is defined, and other parameters that are not required to be kept secret, with the exception of one parameter: the groups are all of order N, where $N = pqr$ is a composite of three large prime numbers. It is assumed that these prime numbers are secret, because the security is derived from an assumption that can be reduced to the assumption that the factorization of N is unknown. Whereas this is not mentioned in [LW11] itself, at the end of a lecture [Lew11], Lewko is asked the same question. She answers that eventually the system should be implemented in a prime-order setting, which is also mentioned in the paper. Earlier the same lecture, it is however briefly mentioned that one of the possible future goals could be to distribute the setup, e.g. with secure multiparty computation (MPC) [BF01].

Nevertheless, this would weaken the security at least to some extent, as it will have to rely upon more security assumptions (as secure MPC might use different security assumptions). Moreover, such multiparty computation protocols often have thresholds where at most half of the participants in the protocol may misbehave in order to keep the value secret. Hence, if the authorities were to use secure MPC to compute N, then this affects the security of the scheme in terms of resilience. As we will see, multi-authority ABE allows the corruption of authorities in the security models, but there is often a limit to the number of authorities that may be corrupted, which is referred to as resilience. In most multi-authority schemes, this limit is almost maximal (i.e. if at least one or two authorities are honest, then the scheme is secure). Obviously, this is not the case when we use secure MPC, because we only need to corrupt half of the authorities in order to break the security of the system, which makes the scheme a lot less resilient and therefore less secure.

In short, we will avoid this problem altogether by just requiring that the order of the groups in the multi-authority ABE scheme is a prime number, which has other advantages as well (which we see in Section 3.8).

3.5.1 Privacy towards the authorities

In decentralized ABE, the keys associated with attributes are distributed by the corresponding authorities. Each authority manages a unique set of attributes, which it does so in a trustworthy manner or not. We already mentioned the notion of corruption in multi-authority schemes, which means that some corrupt authorities may conspire with other corrupt authorities in order to decrypt ciphertexts that they would not be able to decrypt on their own.

But even if authorities are honest in this sense, it does not mean that they cannot infringe upon the privacy of a user in other ways. Consider the following attack: a user with some identifier GID requests keys from certain corrupt authorities such that they collectively manage the attributes “female”, “student” and “computer science”. Then together, they know that the user that corresponds to GID is a female computer science student, which gives the corrupt authorities a significantly better chance at identifying the user behind the identifier GID, whereas knowledge of the separate attributes does not nearly reveal as much information. To avoid this type of attack, we can let the user operate with different pseudonyms at different authorities.

So clearly, using the same global identifier at each authority opens up the possibility to perform a linkability attack: authorities can link different attributes to the same GID, which infringes upon the privacy of the user, at least to some extent. Resistance against this type of attack is also called *authority-unlinkability*, which means that no one can observe that one user requested keys at two different authorities.

In [CC09], Chase and Chow suggest an anonymous key issuing protocol that allows users the use pseudonyms towards authorities such that the user’s privacy is ensured (in the form of authority-unlinkability) while preserving authenticity in the sense that the authority remains certain of the user’s right to the requested key. This can be done by exploiting cryptographic primitives such as commitment schemes and zero-knowledge proofs, but also anonymous credentials (e.g. [Bel+09]).

3.6 Dynamic schemes

One of the main problems in attribute-based encryption was first addressed in [Pir+10], and concerns the revocation of users and/or their keys. Whereas in public-key encryption, we could just blacklist a user and their keys, we cannot simply do this in attribute-based encryption, because the keys correspond to attributes. Blacklisting a users keys would mean that we have to blacklist all the keys in the system that correspond to that very user’s attributes, and then other users cannot use those keys anymore either. This would mean that all keys have to be re-generated and re-distributed, and all of the ciphertexts have to be decrypted and encrypted with the new keys, which is highly impractical.

Yu et al. [Yu+10] mitigate this problem by presenting a method to implement user and attribute revocation by modifying the keys upon each revocation event instead of entirely replacing them with new keys. To this end, Yu et al. use the cryptographic primitive called proxy re-encryption (PRE) [BBS98], which makes it possible for a semi-trusted proxy to ‘re-encrypt’ the ciphertexts. This way, a ciphertext that was encrypted under a certain key can be transformed into a ciphertext that another key (in this case the new key) is able to decrypt. An advantage of this approach is that we do not have to decrypt the old ciphertexts in order to compute the new one. The same holds for the keys, which only need to be updated in order to match the new key. Moreover, we do not have to require the key issuer to be online, but rather allow the re-encryption and key update to be outsourced to a semi-trusted proxy.

In [SSW12], Sahai et al. devise a similar method of implementing revocation in ABE schemes, but it provides a stronger notion of security than the scheme that was proposed in [Yu+10]. They also introduce the notion of revocable storage, which allows an *untrusted* entity to be in charge of the stored encrypted data, and be able to update the ciphertexts in such a fashion that no sensitive data has to be used. In order to do this, Sahai et al. introduce a new procedure called *ciphertext delegation*, which allows a third party to change a ciphertext that was encrypted under a certain policy $\mathbb{A}$ into a ciphertext that is encrypted under a more restrictive policy $\mathbb{A}'$, but only use public information to do this. They also define a property called piecewise key-generation, which can be used in realizing the revocation in ABE.

3.6.1 Directly and indirectly revocable schemes

Attrapadung and Imai [AI09] pointed out that existing revocable ABE systems can be classified in two categories: directly and indirectly revocable ABE. In directly revocable ABE, the list of revoked users is somehow integrated in the ciphertexts by the encryptor. So whereas the revocation itself is in the hands of the trusted third party, the ciphertexts have to be changed by the users, and not by the trusted third party or another semi-trusted proxy. This makes the previously considered approaches that integrate revocation in any ABE construction both indirectly revocable approaches, because they require keys to be updated by an external party, and not the encryptors. Hence, an advantage of direct revocation it that we do not need key updates.

However, one of the disadvantages of direct revocation is that we cannot revoke attributes, which is not desirable in setups such as DECODE, in which attributes are possibly of a temporary nature.

Another disadvantage is that [AI09] puts the effort almost solely on the encryptor of each ciphertext in the system, i.e. the encryptor has to re-encrypt the ciphertexts at each revocation event. Shi et al. [Shi+15] somewhat mitigate this problem by delegating the ciphertext operations to a third party, much like in the indirectly revocable ABE schemes we have considered before. They also provide us with methods to make this delegation verifiable, such that the third party cannot alter the plaintext in the process.

	Direct	**Indirect**
Revocation by	Encryptor	Authority
Key update needed	No	Yes
Attribute revocation	Not trivially	Yes

FIGURE 3.8: Directly versus indirectly revocable ABE

This means that on the one hand, we prefer directly revocable ABE over its indirect counterpart, because it would require a lot less communication in the system from the key generation authority's side. In fact, if the act of revocation were in the hands of an external party, the KGA would not have to do anything at all. But on the other hand, the lack of a trivial way to implement attribute revocation in this direct setting is problematic in settings such as DECODE, so we would have to find another way to do this. Because there are not many ways to do this, this would probably lead to the same or similar solutions as the ones used in indirect revocation. Also, because indirect revocation uses time frames in which keys associated with certain attributes are valid, and DECODE also defines a lifespan on each attribute, it seems that indirect revocation is more compatible with DECODE than direct revocation.

Hence, because of the explicit requirements of DECODE regarding revocation, and the lack of such explicit limitations on the key generation authority in terms of communication during a revocation event, we prefer indirect revocation over direct revocation, even though it might be more costly on the KGA's side.

Remark: Note that we might want the revocation of users and attributes to be done by an external party. This party communicates to the key generation authorities whenever such a revocation event has occurred and which attribute keys need to be renewed. Because, in DECODE, this will probably be realized in combination with attribute-based credentials, we might want to encourage the communication between the party that is responsible for attribute revocation in the ABC setting and the key generation authorities in the ABE setting. Especially in cases when immediate revocation is in order (e.g. when such an event occurs long before the expiration date is due and the revocation is urgent), we might want to enforce a quicker update of keys.

3.6.2 Definition of dynamicity

Intuitively, dynamic ABE is supposed to enjoy certain properties that one typically thinks of as dynamic. We could think of properties such as user and attribute revocation (which we already determined is only possible in indirectly revocable ABE), but also whether an attribute authority can easily join the scheme or whether the number of attribute authorities is fixed after the setup. Another feature that we could consider is whether attributes can be created on the fly in the system, or whether they are fixed after the setup, which depends on whether ABE scheme is a small or large universe construction. Because all of these properties are desirable in ecosystems such as DECODE, we will tailor our definition of dynamic to include these properties.

Definition 17 (Dynamic ABE) *An ABE scheme is dynamic if it is a large universe construction, and if attribute and user revocation are possible. If the ABE scheme is also decentralized, we require that attribute authorities can join the scheme at any moment, even if this is after the setup phase. In particular, this means that the global setup does not generate implicitly secret parameters.*

3.7 Storage and computational costs

Another important factor in cryptosystems are the computational costs, but also the storage costs. We will be considering the possible trade-offs in key and ciphertext sizes, as well as key generation, encryption and decryption operations.

3.7.1 Private key and ciphertext sizes

In attribute-based encryption, the sizes of the keys and/or the ciphertexts may depend on the number of attributes. For instance, the ciphertext may depend the number of attributes that the encryptor uses in the ciphertext, or the size of the key may depend on the number of attributes that a user possesses. For extremely complex access structures, this tends to blow up ciphertexts, and for large sets of attributes, the key storage costs become very expensive.

Whereas most ABE schemes have both key and ciphertext size linear in the number of attributes (e.g. [BSW07]), some are linear in the size of the attribute universe (e.g. [NYO08]). Some manage to make the system such that the ciphertexts and secret keys are of constant size (e.g. [Emu+09]), some have constant-size ciphertexts but have key lengths that are linear in the size of the attribute universe (e.g. [CZF11]), and some have constant-size ciphertexts, but at the cost of the key length being linear in the number of attributes that the user possesses (e.g. [Att+12]).

Depending on the setting, it might be beneficial to use either short keys (when the key storage space is limited, for instance), or short ciphertexts (when the number of ciphertexts is high, and we still allow complex access structures, but the storage space is relatively small).

3.7.2 Key generation, encryption and decryption

Somewhat related to the size of keys and ciphertexts is the computational cost of key generation, encryption and decryption algorithms, which might be dependent on the number of attributes as well, and to some extent also on the size of the keys and ciphertexts. In a similar fashion, we might wish to minimize the number of pairings and exponentiations, which are typically the most expensive operations, that are necessary to compute or decrypt a ciphertext, especially if the operations are executed on devices that do not support complicated operations such as pairing operations, or on devices that do support them, but it would simply cost too much time to be practical.

However, there does not seem to be a trade-off in the computational cost of the key generation, encryption and decryption in the same fashion as in the storage costs of the key and ciphertext. Rather, the computational cost seems to be strongly influenced by the structure of the ABE scheme. For instance, during the decryption, some schemes require for each attribute that satisfies the access policy one or more pairing operations (e.g. [BSW07]), but in other schemes the secret keys and ciphertexts can be combined in a certain way before requiring a pairing operation (e.g. [AC16]). In these schemes, we only need a fixed number of pairing operations. In the key generation and encryption algorithms, we typically do not need any pairing operations, regardless of the ABE scheme that we are considering.

3.8 Group order

Another influence on the storage and computational costs is the group order. In almost all ABE schemes, we either use prime numbers p or composite numbers $N = p_1p_2p_3$ as group order. Here, p and N take the security parameter λ into account, e.g. p is at least λ bits. Because ABE schemes in a composite order setting are implicitly secure under the assumption that given such N, it is computationally infeasible to find primes p_1, p_2, p_3 such that $N = p_1p_2p_3$, we have to require that p_1, p_2, p_3 are large enough primes such that we cannot factorize N. However, the question is how we obtain the same level of security as in the prime-order setting.

In practice, the security level is based on the best attack that is known to solve the problem. This is tested by Guillevic in [Gui13] as well, and it turns out that a composite of three primes only needs to be a couple of bits longer than a composite of two primes (and therefore prime p, which has length λ) would need to be, which makes the difference in storage costs of composite order and prime order ABE almost minimal. This does not mean that the consequences of choosing a group of composite order are minimal. On the contrary, Guillevic has also shown that pairing operations on elements in a group of composite order perform much worse than in a group of prime order, depending on the security parameter, varying from one to two orders of magnitude. So from the computational point of view, we prefer prime order groups over composite order groups.

Moreover, we had already argued in Section 3.5 that in the decentralized setting it would be beneficial to use prime order groups, because otherwise we would need to use secure multiparty computation in order to compute such group order N without leaking the prime factorization, which weakens the security of the system. Now, the only reason why we would prefer the composite order setting is that it is much easier to prove full security in the composite order setting than in the prime order setting, whereas in the prime order setting, we often have to be satisfied with proving a weaker notion of security. Nevertheless, the advantages of prime order groups have resulted in interesting research on how to convert a composite order ABE system into a prime order setting (e.g. [Fre10]), and other research on how to generically construct fully secure ABE systems in prime order setting (e.g. [CGW15; AC16; AC17b]).

3.9 Privacy of access policies

Until now, we have assumed that CP-ABE ciphertexts include the access policies explicitly. However, they might reveal some information about the encryptor. For instance, if a user in the system consistently uses a very specific access policy, then all of the messages are linkable to that user. Moreover, the access policy might also leak some information regarding the user, for example, because the messages are always encrypted with the following policy: Student $\wedge$ ComputerScience. The messages are sent to computer science students, which makes it likely that the sender is a professor at the computer science department.

In other words, in some situations we wish to keep the access policies hidden or private as well. Basically, there are two ways to do this: either we integrate the policy in the encryption and leave the access policy out of the ciphertext completely. In this case we only know whether we satisfy this unknown policy by attempting to decrypt it. If the result is gibberish, then the only thing we learn about the policy is that we did not satisfy it, and if the result does make sense, then we learn that we did satisfy the policy. The alternative is that we include the access policy in the ciphertext, but in a hidden way, i.e. the policy is expressed in hidden attributes that can only be revealed if the user is in possession of the attribute.

In literature, a scheme that does not require the access policy to be public is called *attribute-hiding*[1] [KSW08], i.e. if we do not satisfy the policy, then we learn nothing about the policy at all, and if we do satisfy the policy, we only learn that our own attributes satisfy the policy. Nishide et al. [NYO08] devised a CP-ABE scheme that satisfies this strong notion of security, but they only allow conjunctions in the access policy, and the set of values that each attribute can assume has to be finite and known. Qian et al. [QLZ13] devised another attribute-hiding CP-ABE scheme, but allow the access policies to be more expressive than [NYO08]: they allow both conjunctions and disjunctions on multivalued attributes.

The biggest drawback of these schemes is, ironically, that the policy is completely hidden, and therefore only decryption will show whether we have access to a certain ciphertext. In very large systems, this is highly impractical, as it would force us to decrypt every ciphertext in the system in order to find out if we have access. Rather, we use a slightly weaker notion of security, which we will refer to as *partial attribute-hiding*, in which we allow the attributes and the policy to be visible, but hidden in the

[1]Technically, the official definition of attribute-hiding also includes that the plaintext is hidden, as opposed to *payload-hiding*, which only requires the plaintext to be hidden and not the attributes.

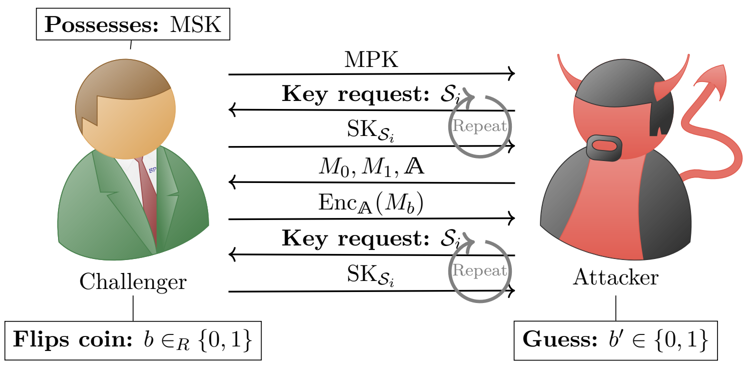

Possesses: MSK
MPK
Key request: $\mathcal{S}_i$
$\mathrm{SK}_{\mathcal{S}_i}$
Repeat
$M_0, M_1, \mathbb{A}$
$\mathrm{Enc}_{\mathbb{A}}(M_b)$
Key request: $\mathcal{S}_i$
$\mathrm{SK}_{\mathcal{S}_i}$
Repeat
Challenger
Attacker
Flips coin: $b \in_R \{0,1\}$
Guess: $b' \in \{0,1\}$

FIGURE 3.9: Security model for ciphertext-policy attribute-based encryption

sense that we can only uncover the attributes in the access policy that match with ours. Then, we do not learn all attributes that are listed in the access policy, but only the ones that we possess ourselves, and then we can conclude instantly whether we can decrypt the ciphertext or not without having to attempt (the much more expensive) decryption. These tactics are used in [Zho+18], in which Zhong et al. exploit the properties of bilinear groups and pairings in order to find the matching attributes. The way in which this is done is fairly generic, and can be applied to any ABE scheme that is proven secure under the DBDH assumption (see Assumption 13).

3.10 Security models

Up till this point, we have discussed attribute-based encryption without discussing the most important part: the security. Much like in regular public-key cryptography, we distinguish between chosen-plaintext attacks (CPA) and chosen-ciphertext attacks (CCA). A CPA-security game in public-key encryption only asks the challenger for a public key associated with some secret key, such that the attacker can encrypt (or query) a polynomially bounded number of plaintexts. When the attacker has collected ‘enough’ of such pairs of plaintexts and ciphertexts, the attacker sends two distinct messages M_0 and M_1 to the challenger. Then the challenger flips a coin, i.e. $b \in_R \{0,1\}$, and sends M_b back encrypted. Then we allow the attacker to query even more plaintexts before he makes a guess. The scheme is said to be *adaptively* CPA-secure if the attacker cannot guess b with non-negligible advantage over random guessing.

3.10.1 Full security

In a similar fashion, we define adaptive or *full* security for attribute-based encryption, but instead of querying over plaintexts, we query secret keys associated with access structures (in the KP-ABE setting) or sets of attributes (in the CP-ABE setting), which ensures that an ABE scheme is collusion resistant. In Fig. 3.9, we have depicted such a security game for CP-ABE. We proceed to give a formal definition of full security CP-ABE (but the definition of full security is similarly defined for KP-ABE).

Definition 18 (Full or adaptive CPA-security for CP-ABE) *We define the game between challenger and attacker as follows:*

- ***Setup:*** *The challenger runs the* Setup *algorithm and sends the public parameters* MPK *to the attacker.*
- ***Query phase I:*** *The attacker queries the challenger for private keys corresponding to sets of attributes* $\mathcal{S}, ..., \mathcal{S}_{n_1}$.
- ***Challenge:*** *The attacker generates two messages* M_0 *and* M_1 *of equal length, together with an access structure* $\mathbb{A}$ *such that none of the queried sets of attributes* $\mathcal{S}_i$ *satisfy* $\mathbb{A}$. *The challenger flips a coin, i.e.* $b \in_R \{0, 1\}$ *and encrypts* M_b *under* $\mathbb{A}$. *It sends the resulting ciphertext to the attacker.*
- ***Query phase II:*** *The attacker queries the challenger for private keys corresponding to sets of attributes* $\mathcal{S}_{n_1+1}, ..., \mathcal{S}_{n_2}$ *with the restriction that none of the sets* $\mathcal{S}_i$ *satisfy access structure* $\mathbb{A}$.
- ***Guess:*** *The attacker outputs a guess* b' *for* b.

The advantage of the attacker is defined as $\Pr[b' = b] - \frac{1}{2}$. *A ciphertext-policy attribute-based encryption scheme is fully (or adaptively) secure if all polynomial-time attackers have at most a negligible advantage in this security game.*

3.10.2 Selective security

In practice, proving full security proved itself to be quite the bottleneck. It was not until 2010 that Lewko et al. [Lew+10] presented the first fully (CPA-)secure ABE scheme. Before this, there have been various ABE schemes (e.g. [SW05; Goy+06; BSW07; CN07]) that were proven *selectively* secure, which is a weaker notion of security. The difference between full security and selective security is that the attack model of selective security is more restricted than the attack model that we just discussed. Instead of letting the attacker choose the structure that he is going to attack in the challenge phase, we let the attacker announce it before the setup starts, which we call the initialization phase.

This allows the challenger to take the structure that will be attacked into account in generating the parameters. This may sound like the mathematical equivalent of being warned by an arsonist that is going to set your house on fire in a week, and you take the appropriate measures against such threats by installing several smoke detectors and fire extinguishers at different places in the house. Then, when the attacker comes, you are prepared against the attack. Of course, you assume that the attacker will set your house on fire, but it is also possible that another attacker, whose proclivities are slightly different from the arsonist's, will break into your house and steal your brand new television. In this case, we can only hope that the fire extinguisher can also be used in a creative way to subdue the intruder. So even though this model is definitely weaker than full security (in which case we are prepared for any attack), the selective security model still offers some useful notion of security.

Whereas almost all of the central authority ABE schemes are proven selectively or fully secure, there is a handful of schemes that is proven secure under a slightly different model. Schemes such as proposed in [AC16] by Agrawal and Chase use the security model that was proposed by Chen and Wee in [CW14b] and is slightly weaker than the full security model, but stronger than the selective security model, and is therefore called the *semi-adaptive* security model. Instead of putting the initialization step before the setup (like in the selective case), they put the initialization step right after the setup phase (and before the first query phase), which means that the challenger cannot tailor the parameters to the attacked structure. However, the attacker has to take the attacked structure into account during both query phases and cannot choose a structure based on previous queries (like in the adaptive case).

3.10.3 Security for multi-authority ABE

In multi-authority attribute-based encryption, we obviously cannot use the same model as the one that we use in the central authority case. Not only are the keys generated by the separate authorities, and therefore the attacker has to request the keys from each separate authority, but we also have take into account that the attacker can potentially corrupt some of those authorities. Also, the setup is split into two parts, the global setup and the authority setup.

Like in the central setting, we can make key queries in an adaptive or a selective fashion. In the same manner, we can corrupt key authorities in an adaptive or a non-adaptive (also called static) fashion, i.e. in-between key queries or strictly before, respectively. To the best of our knowledge, there are no existing multi-authority ABE schemes that allow for adaptive corruption of authorities in the security model[2]. Instead, the attacker is only allowed to corrupt statically, i.e. commit to a list of corrupt users before the query phase of the attack starts. Obviously, we would prefer to also prove security against adaptive corruption, because in practice, some attacker might also be able to corrupt authorities after key queries are made, and not only before. However, as this seems quite possible, it has not happened that a security proof also accounted for adaptive corruption, which means that we will have to focus on static corruption only.

In general we observe in Tab. 3.1 that it is the case that whenever the security model makes the key queries adaptively conform the full security model, the corruption of the authorities happens between the global and the authority setup (e.g. [LW11; RD13; RW15]). The attacker is also allowed to generate public parameters for these corrupted authorities (and the challenger generates the parameters of the honest authorities). Then during the key query phases, the key requests that are sent to the challenger must also take the corrupted authorities into account, such that the corrupted authorities and the sets of attributes that are queried do not satisfy the access structure (in the CP-ABE case).

In the case that keys are made non-adaptively, so conform the selective security model, the corruption of the authorities happens in the initialization step. For the schemes that we investigated, it is also the case that the public and secret parameters are almost always generated by the challenger, whether the authority is corrupt or not. Then the secret keys that belong to the corrupt authorities are also shared with the attacker. This is unlike the full security case, in which we also let the attacker generate the keys of the corrupt authorities.

We have found one multi-authority scheme that does not satisfy either of these two ‘models’, and that is the scheme that was devised by Jung et al. [Jun+13]. Their security model is the same as in the central authority case, and they argue that compromising all but two authorities will not result in the leakage of sensitive information.

Another security aspect that is sometimes discussed in MA-ABE is the anonymity from the user towards the authorities. As we already mentioned, in [CC09], Chase and Chow have devised a generic anonymous key issuing protocol. Moreover, they define a security model to test the security of the protocol, which considers both the possibility of a corrupt user and a corrupt issuer. In [QLZ13; Han+15], blind key

[2]The scheme in [Liu+11] claims to prove security in a model that does allow adaptive corruption, but the set of corrupt authorities is still announced before the key queries are made.

Scheme	Security	Corruption	User anonymity
[Cha07]	Selective	Before global setup	No
[CC09]	Selective	Before global setup	Yes
[LW11]	Full	After global setup	No
[Li+13]	Selective	Before global setup	No
[RD13]	Full	After global setup	No
[Jun+13]	Selective	-	Yes
[QLZ13]	Selective	Before global setup	Yes
[Han+15]	Selective	Before global setup	Yes
[Qia+15]	Selective	Before global setup	Yes
[RW15]	Full	After global setup	No
[Zho+18]	Selective	Before global setup	No

TABLE 3.1: Security of eleven multi-authority schemes

generation protocols are designed that use commitment schemes in order to generate the key. They prove security of the protocol by saying that it should be leak-free and selective-failure blind, which are properties that are introduced in [GH07] and [Han+12]. A key generation protocol is leak-free if a user cannot gain anything from behaving dishonest in a protocol with an honest authority, and it is selective-failure blind if an authority cannot gain information regarding the user's identifier.

3.10.4 Security against chosen-ciphertext attacks

Previously, we have only considered security models for chosen-plaintext attacks. However, in practice we also require the a cryptosystem to be resistant against chosen-ciphertext attacks. Much like there are generic ways to convert a public-key cryptosystem that is resistant against adaptive chosen-plaintext attacks into a cryptosystem that is also resistant against adaptive chosen-ciphertext attacks (e.g. [RS91; Sah99]), there are also generic ways to convert a CPA-secure ABE scheme into a CCA-secure ABE scheme. One of these generic ways was formulated by Canetti et al. in [CHK03], and is often referred to in other papers when it comes to achieving CCA-security (e.g. [Goy+06; BSW07]). This conversion basically introduces a 'new' key pair for each ciphertext that is created such that we cannot simply compute a new ciphertext from an old ciphertext without knowing the 'new' key pair. But this method can only be applied to delegatable ABE schemes (see Section 3.4).

Because not all ABE schemes are delegatable, Yamada et al. [Yam+11] also introduce conversion methods for schemes that are not necessarily delegatable, but they do require schemes to be verifiable instead. The intuitive idea behind verifiability of ABE schemes is that we can verify whether a ciphertext will decrypt to the same plaintext for secret keys associated with two specific sets of attributes (in ciphertext-policy case) or access policies (in the key-policy case). Then for verifiable ABE schemes, Yamada et al. let the encryptors sign their ciphertexts with a one-time signature scheme such that for each plaintext, there is a different sign-verify pair that is used to sign and verify the ciphertext. The idea is that the ciphertext will be signed with the signing key, and can be verified with the verification key.

In order to ensure CCA-security, the verification key is embedded in the structure that is used in the encryption of the plaintext, i.e. for key-policy ABE, this means that the set of attributes is extended, and for ciphertext-policy ABE, the access policy is extended. In order to extend these, we expand the universe of attributes with some dummy attributes that we use to encode the verification key. For these dummy attributes, we do not distribute any keys, but instead use the verifiability property to check whether the decryption of the ciphertext yields the same output for the set of attributes (for KP-ABE) or the access policy (for CP-ABE) corresponding to the verification key encoding as it would for a real, authorized set of keys.

3.10.5 Standard model versus random oracle model

In cryptography, there is a difference between cryptographic schemes that can be proven secure in the standard model and the random oracle model. In the random oracle model, we use hash functions as a random oracle, i.e. the outputs are considered to be truly random, and we need them to be random in order to prove the security of the cryptographic scheme. Because, in practice, hash functions are not truly random, this is a stronger assumption than if we do not need a random oracle. Therefore, we consider ABE schemes that are proven secure in the standard model to be more secure than ABE schemes that need a random oracle to be proven secure.

However, note that we sometimes require one of the properties of a hash function to prove security. For instance, in [SW05], a collision-resistant hash function is necessary to map attribute strings to group elements (in the large universe setting) in order to ensure that two different bit strings are not accidentally represented by the same group element. In this case, the hash function does not need to be modeled as a random oracle.

3.11 Choosing properties for ABE for DECODE

For the DECODE project, we want to find a way to impose access control on data that is not necessarily stored on trusted servers. Our goal is to do this in such a way that it meets the requirements and goals of the ecosystem that DECODE tries to implement. For instance, it requires participants to impose fine-grained access control on their private data in terms of attributes, which is what ciphertext-policy ABE does. As we already discussed before, we also want these access policies to be as expressive as possible whilst being monotone. In practice, this is implemented by means of access trees or LSSS matrices. We also established that we need the scheme to be decentralized, and in order to do this in a privacy-friendly way, we use pseudonyms and blind key generation algorithms to issue the keys.

Another feature of DECODE is that anyone can join the system and become part of the network at any times. If we also take into account that users and attributes can be revoked at all times, then we can conclude that within the ecosystem, there is a need for a certain degree of dynamicity that we described in Definition 17.

In order to make the system as scalable as possible, we make use of session keys: users can encrypt data with symmetric encryption (e.g. AES [DR13]), and then encrypt that session key with attribute-based encryption. This idea was used in ABE schemes such as [HW14] and [AC17a], and is called the *key encapsulation method.* In practice, it is very common to use hybrid encryption schemes such as this.

Other considerations regarding the scalability that we want to take into account are the use of prime orders (as opposed to composite orders), as well as computational and storage costs. For instance, the minimization of the number of pairing operations in the encryption and decryption would be desirable.

Moreover, as we already argued in Section 3.7, there is a trade-off to be made between the size of the secret keys and the size of the ciphertexts. In the setting of DECODE, the secret keys are typically stored in the user's wallet (e.g. on the user's device), so whereas we would like this to be of limited size, we would like to argue that minimizing the ciphertext size is of more importance in this particular setup. The first reason for this is that, in general, in large systems in which there is a lot of information flow, we also need a large number of encrypted session keys. So in contrast, the number of ciphertexts is much larger than the number of users and their corresponding keys. So the total storage size would be smaller if we were to minimize the ciphertext size, and not the key size.

The second reason has to do with the fact that, in principle, the keys only have to be sent to the user once (if we ignore the possibility of key updates), and then they are stored in the user's wallet, whereas the ciphertexts will have to be downloaded from e.g. a cloud server at any moment. Because DECODE may be used on any device, e.g. laptop, smartphone, or other IoT devices, it might be desirable to minimize the amount of data that is downloaded. For instance, consider a smartphone user with a limited amount of data that can be downloaded within the subscription. Suppose that we have used security parameter $\lambda = 2048$, and the average user encrypts his messages with ten attributes in the access policy, for some ABE scheme that has $3n + 1$ group elements in the ciphertext for n attributes (see, for instance, [LW11]). Then the ciphertext has a size of 7.68MB. So if a user may download up to one GB per month under his subscription, he can receive roughly four messages per day that are encrypted like this, which is not even considering the accompanying data.

In short, using ciphertexts that are linear in the number of attributes may not be very efficient. Even if we use the schemes in the elliptic curve setting, we can still argue that the ciphertexts may be too large to be implemented in a scheme that is supposed to be scalable. Rather, we would like to use schemes with constant-size ciphertexts such that the number of attributes that are used in the access policy does not matter.

Optionally, we would also like to allow for 'private' access policies, but as mentioned in Section 3.9, this can easily be done in a generic fashion, as well as converting a CPA-secure scheme into a CCA-secure scheme (see Section 3.10.4).

And finally, we would like the encryption scheme to be proven fully secure (against static corruption) in the standard model to optimize the security guarantees that the attribute-based encryption scheme offers. Furthermore, we prefer cryptosystems of which the security is reducible to well-established complexity assumptions such as DBDH or SXDH over ones that reduce to less well-established complexity assumptions.

So in short, we want to find an ABE scheme that satisfies the following properties:

(i) Ciphertext-policy based;

(ii) Monotone, expressive access policies (access trees or LSSS matrices);

(iii) Large universe;

(iv) Decentralized multi-authority based, where users can authenticate towards authorities with pseudonyms;

(v) Dynamic, i.e.

- allows (indirect) revocation of users and attributes;
- allows for authorities to join the scheme;

(vi) constant-size ciphertexts;

(vii) proven fully secure in the standard model against static corruption.

Chapter 4

Comparing Existing ABE Schemes

So far, we have mainly considered the intuitive notion of attribute-based encryption and which properties they may satisfy. However, it makes sense to consider a couple of explicitly defined schemes in order to give us an idea of what is out there, and how secret sharing is used to realize attribute-based encryption. Furthermore, we compare a number of existing ABE schemes by listing which of the previously defined properties they satisfy.

We will start by giving an example of a CP-ABE scheme, that will also suffice as an example of a large universe CP-ABE scheme. After that, we will give an example of a decentralized CP-ABE scheme. We will omit the security proofs, as they tend to be long and rather tedious. After the examples, we will give a list of ABE schemes (most of which are CP-ABE schemes, but there are some KP-ABE schemes that are interesting for other reasons).

4.1 Example: a (large universe) CP-ABE scheme

The first scheme that we will consider is the first CP-ABE scheme that was formulated, namely the one that devised by Bethencourt et al. in [BSW07]. They represent their access structures with access trees, which are a trees of which the leaves correspond with the attributes in the access policy. Then, each node will either represent a conjunction or a disjunction, i.e. the children of the node will represent partial access policies, and they will be joined into a 'new' access policy by taking either the conjunction or the disjunction of the children (see Fig. 4.1 for an example).

The idea is that the secret value is in the root of the tree, and each node that splits into n branches will implement either a $(1, n)$-threshold scheme (in the disjunction case) or a (n, n)-threshold scheme (in the conjunction case), such that the secret will be shared in such a fashion that it can only be retrieved when a user satisfies the policy. These functions can be realized by using Shamir's secret sharing scheme (see Definition 5) in some clever manner. Each node will define a polynomial such that the degree is either 0 or $n - 1$, where n denotes the number of branches of the node. This is done in a top-down fashion, i.e. the children will share the secret value that they inherited from their parent with a polynomial of degree 0 or $n - 1$ (depending on whether they represent a disjunction or a conjunction respectively), which they will share with their own children again.

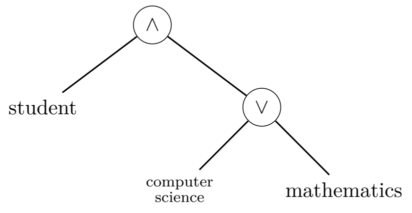


FIGURE 4.1: An access tree that represents the policy "student ∧ (computer science ∨ mathematics)"

This is done as follows: first we number the nodes (we assume that the root has index 0), and we pick some random $s \in_R \mathbb{Z}_p$ as the value that will be shared in the tree. Let n_r be the number of branches of the root node, and $k_r = 1$ if the root represents a disjunction, and $k_r = n_r$ if it represents a conjunction. Then we will generate some random polynomial P_r of degree $k_r - 1$ such that $P_r(0) = s$. We assume that the indices of the children of the root are $1, ..., n_r$. Then the secrets that those nodes will 'inherit' from their parents are points on the polynomial, i.e. $P_r(i)$ will be shared with child with index $i \in [1, n_r]$. Then child with index i will share their secret with their n_i children by determining the threshold $k_i = 1$ or $k_i = n_i$ and generating a polynomial P_i of degree $k_i - 1$ that shares $P_r(i)$, i.e. $P_i(0) = P_r(i)$. Then the same process is repeated for each child and their children until the leaves are reached.

For notation purposes, we will denote parent(x) as the parent of node x, and index(x) as the index of node x. For the leaf nodes y, we define att(y) as the attribute that the leaf y represents. The polynomial that node x generates can be written as $P_x(X) = P_{\text{parent}(x)}(\text{index}(x)) + \sum_{i=1}^{k_x-1} r_{x,i} X^i$, where $r_{x,i} \in_R \mathbb{Z}_p$ are chosen uniformly, and k_x denotes the threshold value of node x. We will denote the access tree as $\mathcal{T}$. If a set of attributes $\mathcal{S}$ satisfies the access policy that $\mathcal{T}$ represents, we denote this as $\mathcal{T} \models \mathcal{S}$.

4.1.1 The construction

Let λ be the security parameter. Let $\mathbb{G}$ be a bilinear group of order p, where p is a prime number of at least λ bits. We define a non-degenerate bilinear map $e : \mathbb{G} \times \mathbb{G} \to \mathbb{G}_T$ and hash function $\mathcal{H} : \{0,1\}^* \to \mathbb{G}$, and we define $\Delta_{i,S}$ as the Lagrange coefficient for element $i \in \mathbb{Z}_p$ and set $S \subseteq \mathbb{Z}_p$, i.e. $\Delta_{i,S} = \prod_{j \in S \setminus \{i\}} \frac{x-j}{i-j}$.

Definition 19 (The [BSW07] scheme) *The ciphertext-policy attribute-based encryption scheme in [BSW07] is defined as follows*[1]*:*

(i) Setup(λ)*: The setup algorithm chooses a bilinear group $\mathbb{G}$ of order p with generator g. Then it picks random $\alpha, \beta \in_R \mathbb{Z}_p$, such that the master secret key is* MSK $= (\beta, g^\alpha)$*, and the master public key is defined as*

$$\text{MPK} = (p, \mathbb{G}, \mathbb{G}_T, e, \mathcal{H}, g, h = g^\beta, h_T = e(g,g)^\alpha).$$

(ii) Encrypt$(\text{MPK}, M, \mathcal{T})$*: The encryption algorithm encrypts a message $M \in \mathbb{G}_T$ under the tree access structure $\mathcal{T}$ by generating polynomials P_x for each node x of $\mathcal{T}$ in the manner that we described above. Recall that the secret that is shared from the root down is the random value $s \in_R \mathbb{Z}_p$. Let Y be the set of leaf nodes of $\mathcal{T}$. Then the ciphertext can be constructed as*

$$\text{CT} = (\mathcal{T}, \hat{C} = M h_T^s, C = h^s, \{C_y = g^{P_y(0)}, C'_y = \mathcal{H}(\text{att}(y))^{P_y(0)}\}_{y \in Y}).$$

(iii) KeyGen$(\text{MSK}, \mathcal{S})$*: The key generation algorithm takes as input the set of attributes $\mathcal{S}$ and outputs a key that corresponds to that set. The algorithm firsts picks a random $r \in \mathbb{Z}_p$ and for each attribute $j \in \mathcal{S}$, it picks random $r_j \in_R \mathbb{Z}_p$. Then the key is computed as*

$$\text{SK} = (\mathcal{S}, D = g^{(\alpha+r)/\beta}, \{D_j = g^r \mathcal{H}(j)^{r_j}, D'_j = g^{r_j}\}_{j \in \mathcal{S}}).$$

[1]We have omitted the delegation algorithm, as this is not relevant to this work.

(iv) Decrypt(CT, SK)*: The decryption algorithm will be performed in a recursive fashion. Let $\mathcal{T}$ be the tree access structure corresponding to the ciphertext* CT*, and $\mathcal{S}$ the set of attributes corresponding to the secret key* SK*. We assume that $\mathcal{T} \models \mathcal{S}$. Let Y' be the set of leaf nodes of $\mathcal{T}$ such that* $\mathrm{att}(y) \in \mathcal{S}$ *for all $y \in Y'$. Then for all $y \in Y'$ we can define* $i = \mathrm{att}(y)$ *and compute*

$$\frac{e(D_i, C_y)}{e(D_i', C_y')} = \frac{e(g^r \mathcal{H}(i)^{r_i}, g^{P_y(0)})}{e(g^{r_i}, \mathcal{H}(i)^{P_y(0)})} = e(g,g)^{rP_y(0)}.$$

For these leaf nodes, we define the output of the function $\mathrm{DecryptNode}(\mathrm{CT}, \mathrm{SK}, y)$ *as $e(g,g)^{rP_y(0)}$. Otherwise the output is defined as $\perp$. Then for each non-leaf node z we define the output of* $\mathrm{DecryptNode}(\mathrm{CT}, \mathrm{SK}, z)$ *as F_z. This value is computed recursively by looking at the values of its children, i.e. let S_z be the set of child nodes such that for all $x \in S_z$, we have found a valid value, i.e. $F_x \neq \perp$, and let $I_z = \{\mathrm{index}(x) : x \in S_z\}$ be the corresponding indices. If $|S_z| < k_z$ holds, then $F_z = \perp$. Otherwise we proceed by computing*

$$F_z = \prod_{x \in S_z} F_x^{\Delta_{\mathrm{index}(x), I_x}(0)} = e(g,g)^{rP_z(0)}.$$

By invoking the DecryptNode *function on the root node r, we compute* $A = \mathrm{DecryptNode}(\mathrm{CT}, \mathrm{SK}, r) = e(g,g)^{rs}$*. Then we can retrieve the plaintext M by computing*

$$\frac{\hat{C}A}{e(C,D)} = \frac{Me(g,g)^{\alpha s} e(g,g)^{rs}}{e(h^s, g^{(\alpha+r)/\beta})} = \frac{Me(g,g)^{\alpha s + rs}}{e(g,g)^{(\alpha+r)s}} = M.$$

4.1.2 Remarks

As we have seen, the scheme is correct in the sense that, for access structure $\mathcal{T}$ and set of attributes $\mathcal{S}$ such that $\mathcal{T} \models \mathcal{S}$, we have the decryption yields the same message M that was used in the encryption. For $\mathcal{S}$ such that $\mathcal{T} \not\models \mathcal{S}$, the decryption algorithm should fail. More importantly, it should not reveal any information regarding M.

The scheme is proven selectively secure in the generic bilinear group model and random oracle model. Whereas the security of most other schemes can be reduced to some problem that is considered to be intractable, the security of this particular scheme is not shown to be reducible to any of such intractable problems, but rather uses random encodings and hashes in the proof.

Finally, we note that the scheme is a large universe construction, because the number of attributes is not polynomially, but exponentially bounded in the security parameter after the setup, as this scheme allows for each possible attribute string to be mapped to a group element in $\mathbb{G}$. Moreover, the master public key only consists of a constant number of elements and is therefore not dependent on the number of attributes.

4.2 Example: a decentralized CP-ABE scheme

In [LW11], Lewko and Waters published a fully decentralized CP-ABE scheme that is provably adaptively secure against static corruption. Unlike the previous CP-ABE scheme, the scheme requires a small universe of attributes, because we define a public parameter for each attribute in the universe. As we had established in Section 3.3, a large universe construction cannot have this feature. Moreover, the groups are not of prime, but composite order. Another thing that is different is the access structure. Whereas the expressiveness is the same, i.e. all monotone access policies can be used in the encryption, the access structure is not represented by an access tree, but rather uses the LSSS matrices that we defined in Section 2.6.1.

Recall that we defined an LSSS access structure as $(\mathbf{A}, \rho)$, where A denotes the $n_1 \times n_2$ share-generating matrix and ρ the function that maps the rows of $\mathbf{A}$ to attributes in the system. A set of attributes $\mathcal{S}$ satisfies the access policy if there is a set of rows of $\mathbf{A}$ such that the corresponding attributes are in $\mathcal{S}$ and the vector $(1, 0, ..., 0)$ is in the span of those rows. We define Υ as the set of indices of the rows that correspond with attributes in $\mathcal{S}$, i.e. $\Upsilon = \{i \in [1, n_1] : \rho(i) \in \mathcal{S}\}$.

As we already mentioned, multi-authority ABE schemes have the trivial disadvantage that the secret keys are not distributed by one central authority and therefore have to tie the keys to an identity in quite a literal fashion in order to make collusion impossible.

Definition 20 (The [LW11] scheme) *The decentralized multi-authority ciphertext-policy attribute-based encryption scheme in [LW11] is defined as follows:*

(i) GlobalSetup(λ)*: In the global setup, we choose a bilinear group $\mathbb{G}$ of order N, where $N = pqr$ is a composite of three primes p, q and r. We also pick a generator $g \in \mathbb{G}_p$. Let $e : \mathbb{G} \times \mathbb{G} \to \mathbb{G}_T$ be a non-degenerate bilinear mapping. We also define a hash function $\mathcal{H} : \{0,1\}^* \to \mathbb{G}$ that maps global identifiers to group elements, which we model as a random oracle. The global parameters are*

$$\mathrm{GP} = (N, \mathbb{G}, \mathbb{G}_T, g, e, \mathcal{H}).$$

(ii) AuthoritySetup(GP)*: For each attribute y_k that is controlled by authority $\mathcal{A}_k$, the authority chooses two random exponents $\alpha_{y_k}, \beta_{y_k} \in_R \mathbb{Z}_N$ and keeps* $\mathrm{MSK}_{\mathcal{A}_k} = \{\alpha_{y_k}, \beta_{y_k}\}_{y_k}$ *as the secret key. The public key is defined as*

$$\mathrm{MPK}_{\mathcal{A}_k} = \{e(g,g)^{\alpha_{y_k}}, g^{\beta_{y_k}}\}_{y_k}.$$

(iii) Encrypt$(\mathrm{GP}, \{\mathrm{MPK}_{\mathcal{A}_k}\}_{\mathcal{A}_k}, M, (\mathbf{A}, \rho))$*: For the encryption, we take as input the message $M \in \mathbb{G}_T$ and access structure $(\mathbf{A}, \rho)$, as well as the global parameters and the public keys of the relevant authorities. Then we pick random elements $s, v_2, ..., v_{n_2} \in_R \mathbb{Z}_N$ such that we can define vector $\mathbf{v} = (s, v_2, ..., v_{n_2})$. Let $\lambda_i = A_i \mathbf{v}$, where A_i denotes the i-th row of $\mathbf{A}$. We also pick $w_2, ..., w_{n_2} \in_R \mathbb{Z}_N$ such that we can define vector $\mathbf{w} = (0, w_2, ..., w_{n_2})$. Then we define $\omega_i = A_i w$ for each row A_i. We also pick random $r_i \in_R \mathbb{Z}_N$ for each row. Then the ciphertext is computed as*

$$\mathrm{CT} = ((A, \rho), C_0 = M \cdot e(g,g)^s, \{C_{1,i} = e(g,g)^{\alpha_{\rho(i)} r_i + \lambda_i},$$
$$C_{2,i} = g^{r_i}, C_{3,i} = g^{\beta_{\rho(i)} r_i + \omega_i}\}_{i \in [1, n_1]}).$$

(iv) KeyGen$(\mathrm{GP}, \mathrm{GID}, i, \mathrm{MSK}_{\mathcal{A}})$*: To create a key for a user with global identifier* GID *for attribute y_k managed by authority $\mathcal{A}_k$, the authority computes*

$$\mathrm{SK}_{y_k, \mathrm{GID}} = g^{\alpha_{y_k}} \mathcal{H}(\mathrm{GID})^{\beta_{y_k}}.$$

(v) Decrypt$(\mathrm{GP}, \mathrm{CT}, \{\mathrm{SK}_{y,\mathrm{GID}}\}_{y \in \mathcal{S}})$*: Let* CT *be the ciphertext that we want to decrypt, $(\mathbf{A}, \rho)$ the access structure under which it is encrypted, where $\mathbf{A}$ is an*

$n_1 \times n_2$ matrix, $\mathcal{S}$ the set of attributes for which the user has a secret key and GID *the global identifier of the user that wants to decrypt* CT. *First, we compute $\mathcal{H}(\mathrm{GID})$, and determine $\mathrm{Y} = \{i \in [1, n_1] : \rho(i) \in \mathcal{S}\}$. Let $\{c_i\}_{i \in \mathrm{Y}}$ be elements in $\mathbb{Z}_N$ such that $\sum_{i \in \mathrm{Y}} c_i A_i = (1, 0, ..., 0)$. Then the decryptor computes for each $i \in \mathrm{Y}$:*

$$C_{1,i} \cdot e(\mathcal{H}(\mathrm{GID}), C_{3,i}) / e(\mathrm{SK}_{\rho(i),\mathrm{GID}}, C_{2,i}) = e(g,g)^{\lambda_i} e(\mathcal{H}(\mathrm{GID}), g)^{\omega_i},$$

and combines them by computing

$$\prod_{i \in \mathrm{Y}} (e(g,g)^{\lambda_i} e(\mathcal{H}(\mathrm{GID}), g)^{\omega_i})^{c_i} = e(g,g)^s.$$

Note that $\prod_{i \in \mathrm{Y}} e(\mathcal{H}(\mathrm{GID}), g)^{c_i \omega_i}$ cancels out because $\sum_{i \in \mathrm{Y}} c_i \omega_i = \sum_{i \in \mathrm{Y}} c_i A_i \mathbf{w} = 0$. Finally, we retrieve plaintext M by computing

$$M = C_0 / e(g,g)^s.$$

4.2.1 Remarks

Correctness of the scheme above follows from the fact that $e(g^{\alpha_{\rho(i)}} \mathcal{H}(\mathrm{GID})^{\beta_{\rho(i)}}, g^{r_i}) = e(g^{\alpha_{\rho(i)}}, g^{r_i}) \cdot e(\mathcal{H}(\mathrm{GID})^{\beta_{\rho(i)}}, g^{r_i})$ and

$$\begin{aligned}
&C_{1,i} \cdot e(\mathcal{H}(\mathrm{GID}), C_{3,i}) / e(\mathrm{SK}_{\rho(i),\mathrm{GID}}, C_{2,i}) \\
&= e(g,g)^{\alpha_{\rho(i)} r_i + \lambda_i} \cdot e(\mathcal{H}(\mathrm{GID}), g^{\beta_{\rho(i)} r_i + \omega_i}) / e(g^{\alpha_{\rho(i)}} \mathcal{H}(\mathrm{GID})^{\beta_{\rho(i)}}, g^{r_i}) \\
&= e(g,g)^{\lambda_i} \cdot e(\mathcal{H}(\mathrm{GID}), g)^{\beta_{\rho(i)} r_i + \omega_i} / e(\mathcal{H}(\mathrm{GID}), g)^{\beta_{\rho(i)} r_i} \\
&= e(g,g)^{\lambda_i} \cdot e(\mathcal{H}(\mathrm{GID}), g)^{\omega_i}.
\end{aligned}$$

The rest is fairly straightforward, because we have $\sum_{i \in \mathrm{Y}} c_i \lambda_i = \sum_{i \in \mathrm{Y}} c_i A_i \mathbf{v} = s$ and $\sum_{i \in \mathrm{Y}} c_i \omega_i = \sum_{i \in \mathrm{Y}} c_i A_i \mathbf{w} = 0$, which we can use to compute

$$\prod_{i \in \mathrm{Y}} (e(g,g)^{\lambda_i} e(\mathcal{H}(\mathrm{GID}), g)^{\omega_i})^{c_i} = e(g,g)^{\sum_{i \in \mathrm{Y}} c_i \lambda_i} e(\mathcal{H}(\mathrm{GID}), g)^{\sum_{i \in \mathrm{Y}} c_i \omega_i} = e(g,g)^s.$$

We also have to point out that in the access policies, each attribute may only be used once, which is an unfortunate feature of this scheme. The reason for this is that in the security proof, there needs to be a restriction on ρ in the sense that it has to be injective, otherwise there is not 'enough randomness' in the group elements. Whereas this problem can be solved in a relatively simple way, it makes the scheme a little less efficient. The solution requires each authority to generate another set of secret and public keys for each possible occurrence of the attribute in any policy.

The security of this scheme is derived from the dual system encryption methodology that was introduced by Waters in [Wat09], which leads to encryption schemes that are fully secure in the standard model under simple and established assumptions such as the decisional bilinear Diffie-Hellman and the decisional linear assumptions. We will see more about this methodology in Appendix A.

4.3 The comparison of forty-two schemes

At the end of Chapter 3, we had set out a couple of properties that we wish to implement in our ABE scheme. Naturally, we would like to consider the schemes that are already out there, and which properties they satisfy.

4.3.1 First set of properties

Because we have a lot of properties to take into account, we will first distinguish between a selective number of structural and non-security related properties, namely

(i) Key-policy (KP) versus ciphertext-policy (CP);

(ii) Small versus large universe;

(iii) Monotonic versus non-monotonic;

(iv) Expressiveness of access policies, we distinguish between:

- Threshold function;
- CNF (conjunctive normal form) with wildcards);
- Conjunctions ($\wedge$) on positive and negative attributes ($\pm$) (or wildcards);
- Conjunctions ($\wedge$) on multi-valued (mv) attributes (with or without wildcards);
- Conjunctions ($\wedge$) and disjunctions ($\vee$) on multi-valued attributes;
- Access trees;
- LSSS matrices;

(v) Central authority versus multi-authority;

(vi) Allows *indirect* revocation of users;

(vii) Whether the ciphertext is of constant size;

(viii) Prime versus composite order.

Note that we have only included *user* revocation as a property, and not *attribute* revocation. However, as we had already mentioned, with indirect revocation we can implement attribute revocation in a similar fashion as user revocation.

4.3.2 Second set of properties

In the second comparison of the schemes, we compare the schemes with regard to their security properties. We compare the following notions of security:

(i) Selective versus full security;

(ii) Security against chosen-plaintext attacks (CPA) versus chosen-ciphertext attacks (CCA);

(iii) Random oracle versus standard model;

(iv) Complexity assumption that the security of the scheme reduces to, for instance:

- Decisional bilinear Diffie-Hellman (DBDH) (see Assumption 13);
- Symmetric external Diffie-Hellman (SXDH) (see Assumption 14);
- Generic (bilinear) group heuristic (G(B)GH) (e.g. [BSW07]);
- Decisional linear (DLIN) [BBS04];
- Subgroup, which is only used in a composite setting (e.g. [LW11]);
- q-type assumptions, where the number of queries that may be made in the attack model is bounded by q;
- Other assumptions that are not specified in this work and have a less-established status.

4.3.3 Methodology and notations

In the comparison of the schemes, we have used several strategies that might result in the inclusion or exclusion of certain schemes that other works might have excluded or included while analyzing the same properties.

For instance, we have only indicated that a scheme satisfies a certain property if the authors of the work in question have *explicitly* defined and implemented said property in the scheme. So, for example, only noting in the work that the scheme can be decentralized is not sufficient to satisfy this property for the sake of this comparison (e.g. [CZF11]). Another frequently recurring example is noting that a scheme is easily extendable to resist chosen-ciphertext attacks by using the generic constructions that we mentioned in Section 3.10.4 (e.g. [Goy+06; BSW07]).

The reason for this is that simply mentioning that a certain property can be implemented does not guarantee that it can be implemented in the way that we defined in this work or that we considered to be suitable for DECODE, whereas the explicit definition guarantees that we can assess the scheme with our own interpretation of the property in mind.

After evaluating all forty-two schemes with regard to the sets of properties, we analyze which schemes satisfy the most properties of the first set. For the second set, we will simply analyze which schemes are proven fully secure. In our 'second round of analysis' we determine of the fully secure schemes that did not belong to the first set of selected properties whether their lack of certain properties is 'redeemable' or not. For instance, a small universe construction would not fit in this category, because the system has to be structurally changed in order to be a large universe construction. Similarly, access policies that are not represented by access trees or LSSS matrices are not redeemable either, because the access structure is also part of the structure of an ABE scheme. The same holds for the size of the ciphertexts, but we will not discriminate on this property, because there is only one scheme that has both constant-size ciphertexts and expressive policies, and we definitely prefer expressive policies over constant-size ciphertexts in DECODE.

From the set of schemes that we select based on this analysis, we discuss which one would suit the DECODE setting the best based on certain trade-offs.

For notation purposes, we have indicated the wildcards with an asterisk in Tables 4.1 and 4.2. The properties that are put in purple indicate that they satisfy the properties that we have chosen for our specific setup in Section 3.11.

4.4 Discussion

As we can see in Tables 4.1, 4.2 and 4.3, there are no schemes that satisfy all (or even all but one) properties that we want to impose on our ABE scheme.

However, there are six schemes that satisfy six of the eight properties that we considered in Tables 4.1 and 4.2, namely:

(i) [BSW07], which is not decentralized, and does not have constant-size ciphertexts;

(ii) [Jun+13], which implements no indirect revocation but direct revocation, and it does not have constant-size ciphertexts;

(iii) [Qia+15], which has a small universe of attributes, and does not have constant-size ciphertexts;

(iv) [RW15], which does not support user revocation, and has no constant-size ciphertexts;

(v) [AC16], which is not decentralized and does not support user revocation;

(vi) [Zho+18], which has a small universe of attributes and does not have constant-size ciphertexts.

Of these six schemes, only one is proven fully secure (against static corruption, in the random oracle model), namely [RW15], and one is proven semi-adaptively secure (and fully secure in [AC17b]) in the standard model, namely [AC16].

To broaden our scope of possible schemes that are fully secure (in either the random oracle or standard model), we can also consider all fully secure schemes and determine whether their lack of desirable properties is 'redeemable' or not.

As we have shown, there are only eight schemes that are proven fully secure. We will also indicate which are redeemable and which are not. As it turns out, almost all of the schemes are small universe constructions, i.e.

(i) [Lew+10; LW11; SSW12; RD13; Lai+13; Den+14] are small universe constructions, hence irredeemable;

(ii) [RW15] may be redeemable, because has a large universe of attributes, and expressive policies;

(iii) [AC17a] may be redeemable, because has a large universe of attributes, and expressive policies.

If we only want to allow (semi-)adapatively secure schemes, this means we have narrowed down our set of possible schemes to three, namely [RW15], [AC16] and [AC17a], which is a very small group of schemes. All of these are large universe CP-ABE schemes, with monotone access policies that are represented by LSSS matrices and the groups are of prime order, and proven (semi-)adaptively secure.

If we extend our search parameters a tiny bit by also allowing selectively secure schemes, and further requiring that the schemes are large universe CP-ABE schemes, with expressive, monotone access policies and the groups are of prime order, then we can extend our selection by three: [BSW07], [Jun+13], and [RW13]. Note that none of these have a significant advantage regarding the computational or storage costs. The only advantage that any of these schemes have over the other schemes is that [Jun+13] implements user revocation (but does not support attribute revocation yet) and allows to user to act anonymously towards the authorities. Hence, it is one of the most complete schemes, even if it is only selectively secure.

Then, objectively speaking, the scheme as set out in [AC17a] might be the best one to choose regarding the computational and storage costs, as it was created to optimize the speed of the key generation, encryption and decryption algorithms, whereas the storage costs are roughly the same as in the other schemes (with the exception of [AC16]). However, we think that the scheme in [AC17a] might not be too easy to decentralize without breaking some of the security guarantees.

4.5 Concluding remarks

It appears as though all of the forty-two schemes that we have studied have their own advantages and disadvantages, which makes it very difficult to pick 'one perfect scheme'. If security guarantees and speed are important, then [AC17a] is probably the best choice. However, we think that it will be very hard to decentralize this scheme, so it may arguably not be the best choice for ecosystems such as DECODE, which aims to decentralize the trust.

If security guarantees are not the most important factor, but rather the 'completeness' of the scheme, i.e. the implementation of as many of the properties that we have set out to be the most important in the setting of DECODE as possible, [Jun+13] might be the best choice, because it already has almost all the features that we want to impose on the scheme, except for attribute revocation and the constant-size ciphertexts property, without becoming extremely inefficient. It has also implemented a blind key generation protocol such that users can be anonymous towards the authorities, which was also one of the requirements that we formulated at the end of Chapter 3.

Because we considered the security and decentralization properties to be very important, we feel as though [RW15] is probably the best choice, as it implements almost all of the properties that we have set out in Section 3.11, except for revocation and constant-size ciphertexts. However, it allows for easy re-randomization of ciphertexts, so we suspect that this can easily be used to apply proxy re-encryption techniques, which are used to implement indirect revocation (and does not only allow user but also attribute revocation). In our opinion, this makes [RW15] the best option for DECODE, because it is both usable, efficient, and proven fully secure against static corruption whilst adhering to almost all properties we set out.

4.6 CP-ABE with constant-size ciphertexts

As we have seen in Tab. 4.1 and 4.2, only four of the forty-two schemes that we have evaluated enjoy the constant-size ciphertexts propery. Because there is very little research on constant-size ciphertexts in CP-ABE schemes, and there are, to the best of our knowledge, no multi-authority ciphertext-policy schemes that have constant-size ciphertexts and still allow for expressive access structures, it might be interesting to look a bit further into the [AC16] scheme and the possibility of decentralizing it, and perhaps even consider whether we can implement revocation of some proper kind. Because [AC16] is more of a baking manual than an actual cake that is ready to be devoured, we will have to dive a bit deeper into the background of [AC16], and how we get to the encryption scheme that we have used in our analysis. In Appendix A we will walk through the steps that Agrawal and Chase have described in their work, which will eventually lead to the construction of a semi-adaptively secure CP-ABE scheme with constant-size ciphertexts in Appendix B. Finally, in Appendix C, we attempt to construct the first decentralized CP-ABE scheme with constant-size ciphertexts that allows for expressive access structures, which we try to prove semi-adaptively secure against static corruption.

Scheme	KP/CP	Universe	Monotone?	Access policies	Multi-authority?	Revocation?	Short CT	Order
[SW05] (1)	KP	Small	Yes	Threshold	No	No	No	Prime
[SW05] (2)	KP	Large	Yes	Threshold	No	No	No	Prime
[Goy+06] (1)	KP	Small	Yes	Access trees	No	No	No	Prime
[Goy+06] (2)	KP	Large	Yes	Access trees	No	No	No	Prime
[BSW07]	CP	Large	Yes	Access trees	No	Yes, indirect	No	Prime
[Cha07] (1)	KP	Small	Yes	Threshold	Yes, but with CA	No	No	Prime
[Cha07] (2)	KP	Large	Yes	Access trees	Yes, but with CA	No	No	Prime
[CN07] (1&2)	CP	Small	No	$\wedge$ on $\pm$*	No	No	No	Prime
[OSW07]	KP	Small	No	LSSS matrices	No	No	No	Prime
[NYO08] (1&2)	CP	Small	No	$\wedge$ on mv*	No	No	No	Prime
[Emu+09]	CP	Small	No	$\wedge$ on mv	No	No	Yes	Prime
[CC09]	KP	Small	Yes	Threshold	Yes, decentralized	No	No	Prime
[AI09] (1)	KP	Large	Yes	LSSS matrices	No	Yes, direct	No	Prime
[AI09] (2)	KP	Large	Yes	LSSS matrices	No	Yes, indirect	No	Prime
[Lew+10]	CP	Small	Yes	LSSS matrices	No	No	No	Composite
[Yu+10] (1&2)	CP	Small	No	$\wedge$ on $\pm$*	No	Yes, indirect	No	Prime
[CZF11] (1&2)	CP	Small	No	$\wedge$ on $\pm$*	No	No	Yes	Prime
[Wat11] (1-3)	CP	Small	Yes	LSSS matrices	No	No	No	Prime

TABLE 4.1: Comparison of forty-two schemes

Scheme	KP/CP	Universe	Monotone?	Access policies	Multi-authority?	Revocation?	Short CT	Order
[LW11]	CP	Small	Yes	LSSS matrices	Yes, decentralized	No	No	Composite
[Att+12]	CP	Small	Yes	Threshold	No	No	Yes	Composite
[SSW12]	CP	Small	Yes	LSSS matrices	No	Yes, indirect	No	Prime
[Li+13] (PUD)	CP	Small	No	CNF with *	Yes, decentralized	Yes, indirect	No	Prime
[Li+13] (PSD)	KP	Small	No	CNF with *	Yes, decentralized	Yes, indirect	No	Prime
[RD13]	CP	Small	Yes	Any MAS	Yes, decentralized	No	No	Prime
[Jun+13]	CP	Large	Yes	Access trees	Yes, decentralized	Yes, direct	No	Prime
[Lai+13]	CP	Small	Yes	LSSS matrices	No	No	No	Prime
[QLZ13]	CP	Small	Yes	$\wedge$ and $\vee$ on mv	Yes, decentralized	No	No	Prime
[RW13]	CP	Large	Yes	LSSS matrices	No	No	No	Prime
[Den+14]	CP	Small	Yes	LSSS matrices	No	No	No	Composite
[Shi+15]	KP	Small	Yes	LSSS matrices	No	Yes, direct	No	Prime
[Han+15]	CP	Small	Yes	LSSS matrices	Yes, decentralized	No	No	Prime
[Qia+15]	CP	Small	Yes	Access trees	Yes, decentralized	Yes, indirect	No	Prime
[RW15]	CP	Large	Yes	LSSS matrices	Yes, decentralized	No	No	Prime
[AC16]	CP	Large	Yes	LSSS matrices	No	No	Yes	Prime
[AC17a]	CP	Large	Yes	LSSS matrices	No	No	No	Prime
[Zho+18]	CP	Small	Yes	LSSS matrices	Yes, decentralized	Yes, indirect	No	Prime

TABLE 4.2: Comparison of forty-two schemes (continued)

Remark: Note that the [RD13] scheme allows any monotone access structure (MAS) as access policy, which also includes access trees and LSSS matrices.

Scheme	**Security**	**CPA/CCA**	**RO model?**	**Assumption**
[SW05] (1)	Selective	CPA	No	Other
[SW05] (2)	Selective	CPA	No	DBDH
[Goy+06] (1)	Selective	CPA	No	DBDH
[Goy+06] (2)	Selective	CPA	No	DBDH
[BSW07]	Selective	CPA	Yes	GBGH
[Cha07] (1)	Selective	CCA	No	DBDH
[Cha07] (2)	Selective	CPA	No	DBDH
[CN07] (1)	Selective	CCA	No	DBDH & Other
[CN07] (2)	Selective	CPA	No	DBDH
[OSW07]	Selective	CPA	No	DBDH
[NYO08] (1)	Selective	CPA	No	DBDH & DLIN
[NYO08] (2)	Selective	CPA	No	GGH
[Emu+09]	Selective	CPA	No	DBDH
[CC09]	Selective	CPA	No	DBDH & Other
[AI09] (1& 2)	Selective	CPA	No	DBDH
[Lew+10]	Full	CPA	No	Subgroup
[Yu+10] (1)	Selective	CPA	No	DBDH
[Yu+10] (2)	Selective	CCA	No	DBDH
[CZF11] (1)	Selective	CPA	No	GGH & Other
[CZF11] (2)	Selective	CCA	No	GGH & Other
[Wat11] (1-2)	Selective	CPA	No	Other
[Wat11] (3)	Selective	CPA	No	DBDH
[LW11]	Full	CPA	Yes	Subgroup
[Att+12]	Selective	CPA	No	Other
[SSW12]	Full	CPA	No	Subgroup
[Li+13] (PUD)	Selective	CPA	No	DBDH
[Li+13] (PSD)	Selective	CPA	No	DBDH
[RD13]	Full	CPA	Yes	GBGH
[Jun+13]	Selective	CPA	Yes	DBDH
[Lai+13]	Full	CCA	No	DBDH
[QLZ13]	Selective	CPA	No	DBDH
[RW13]	Selective	CPA	No	q-type
[Den+14]	Full	CPA	No	Subgroup
[Shi+15]	Selective	CPA	Yes	q-type
[Han+15]	Selective	CPA	No	q-type
[Qia+15]	Selective	CPA	No	DBDH
[RW15]	Full	CPA	Yes	q-type
[AC16]	Semi-adaptive	CPA	No	SXDH or DLIN
[AC17a]	Full	CPA	Yes	DLIN
[Zho+18]	Selective	CPA	Yes	Other

TABLE 4.3: Comparison of security of forty-two schemes

Remark: Note that the [AC16] scheme is proven fully (CPA-)secure in the standard model under a q-type assumption in [AC17b].

Chapter 5

Conclusions

In this work, we have analyzed how attribute-based encryption can be used and which types exist. We have seen that there is a wide variety of ABE schemes, and that they come in many flavors and with many useful properties that are suitable in certain setups. For instance, there are schemes that provide expressive access structures, schemes that are decentralized and schemes that are dynamic, or any combination of these properties. In the particular setup of DECODE, we had argued that we need a combination of the following properties:

(i) Ciphertext-policy based;

(ii) Monotone, expressive access policies (access trees or LSSS matrices);

(iii) Large universe;

(iv) Decentralized multi-authority based, where users can authenticate towards authorities with pseudonyms;

(v) Dynamic, i.e.

- allows (indirect) revocation of users and attributes;
- allows for authorities to join the scheme;

(vi) constant-size ciphertexts;

(vii) proven fully secure in the standard model against static corruption.

However, as it turned out, this combination of properties is too high of a standard. The biggest bottleneck in this search was the constant-size ciphertext requirement, as there are not many schemes that satisfy this property in general. Nevertheless, even if we were to eliminate the constant-size property, we had a difficulty in finding a scheme that satisfies all of these properties. There was one scheme that would fit almost all of these properties, namely the scheme created by Rouselakis and Waters in [RW15]. For this scheme, we would only have to implement the dynamicity property in some fashion. We also argued that this is probably feasible by integrating indirect revocation by means of proxy re-encryption. As we have seen, this would provide us with both user and attribute revocation.

In the appendices of this work, we considered generic frameworks for creating attribute-based encryption schemes, which we did in order to construct a ciphertext-policy attribute-based encryption scheme with constant-size ciphertexts. From there, we created a decentralized multi-authority CP-ABE scheme with constant-size ciphertexts, which sacrifices the large universe property of the old scheme, and is extremely inefficient. To the best of our knowledge, this is the first decentralized multi-authority ciphertext-policy attribute-based encryption scheme with constant-size ciphertexts that is provably semi-adaptively secure against static corruption in the standard model. However, it might not be efficient enough in settings such as DECODE, which require the scheme to be scalable, as it is supposed to handle possibly large amounts of users and authorities. Another disadvantage is that the identity consists of $4n_1$ group elements, contrary to other schemes that typically use one element to represent the global identifier.

5.1 Further research

There is still a lot of progress in the area of attribute-based encryption. Especially in more generic encryption schemes, such as predicate and functional encryption, and generic frameworks, such as the ones we have seen in the second part of this work, there is still a lot of research to be done.

However, as we have seen, there is very little research on CP-ABE with constant-size ciphertexts, and even less so in the multi-authority setting. This work attempts to change that. Nevertheless, we have doubts that the scheme as it is would ever be efficient enough, and it still lacks some of the properties that we deem important in settings such as DECODE, for instance, the large universe and other dynamicity properties. In the future, it might be possible to use the ideas that we have formulated in this work in a more efficient setup, and perhaps implement a blind key generation algorithm to ensure the privacy of the users towards the authorities. Moreover, we recommend changes to the system such that it can be made dynamic. Whereas revocation is probably possible to implement in the scheme as the keys are somewhat homomorphic, the construction as it is now is a small universe construction, and authorities cannot join the system without requiring that at least a part of the global setup should be run again.

To this end, we recommend looking into the possibility of letting the parameters correspond with the authorities, e.g. let $\mathbf{g}_{i,j}$ correspond with authority $\mathcal{A}_i$, such that we only require a round of communication in the generation of $\mathbf{g}_0$ and $\mathbf{h}_0$, and potentially $\bar{\mathbf{g}}_0$ and $\bar{\mathbf{h}}_0$. We attempt to solve two problems with this: the lack of efficiency in the old system, because this way, the authorities only need one or two rounds of communication in the generation of $\mathbf{g}_0$ and $\mathbf{h}_0$ (and potentially $\bar{\mathbf{g}}_0$ and $\bar{\mathbf{h}}_0$), and the rest of the parameters can be generated by the authorities that are responsible for certain parameters. This way, any authorities that wish to join can also create their own parameters at any point in the scheme, not just during the setup. Moreover, authorities that manage a set of attributes can add another attribute to their set at any time, because it only needs T to be large enough such that the users cannot forge keys (and it is not additionally required to provide security against corrupt authorities), so this would restore the large universe property of the scheme as well. However, this setup might come with its own difficulties, as it still requires the generation of a master secret key, and other parameters that all depend on one another in some way.

Other than the lack of pseudonyms, large universe and dynamicity properties, the scheme that we created in Definition 35 satisfies all properties that we deemed necessary in DECODE. Moreover, we believe that we can make the scheme secure against chosen-ciphertext attacks with the generic methods as discussed in Section 3.10.4. We can also implement private access policies with the approach as discussed in Section 3.9, though we would have to adapt the approach according to a setting with two different source groups, and perhaps we would also need different or stronger security assumptions than before. Nevertheless, we believe that this makes the scheme somewhat more usable in some setups that do not require dynamicity properties.

Appendix A

Simplifying Security Analysis with Pair Encoding Schemes

In Section 4.6, we already mentioned that [AC16] only provides the building blocks and instructions to create a CP-ABE scheme with constant-size ciphertexts. However, these building blocks are not straightforward, and how they can be combined in such a fashion that it fits the formal description of Definition 16 might be difficult to understand. There are two notions that warrant further explanation and definition in order to truly understand how these schemes are created and why they are secure. These notions are *pair encoding schemes* (PES) and *dual system groups* (DSGs).

Of course, these notions are not solely useful for the purpose of defining our CP-ABE scheme, but provide a generic way to construct new ABE schemes and analyze its security with information-theoretic notions. Proofs of security in ABE schemes are often long and difficult, and might not always reduce to well-established security assumptions. By using pair encoding, we can prove security in a relatively easy fashion under well-established assumptions such as the SXDH assumption.

Note that during this chapter, we will mostly focus on the work of Agrawal and Chase [AC16], so almost all of the definitions are inherited from their work as well. Because their work is quite abstract and difficult to grasp, we provide intuitive explanations of some of the more complicated notions that are discussed in [AC16].

A.1 An intuitive overview

As we have seen, the vast majority of the ABE schemes that have been published is only proven to be selectively secure, and not fully secure. Of the forty-two schemes that we have considered, only eight are proven fully secure (nine if we also count [AC16]). Four of these are ABE schemes in the composite-order setting, and we already established that we preferred prime-order ABE schemes in Section 3.8. Six of the schemes are proven fully secure by using the methodology of [Wat09] that introduces the notion of *dual system encryption*.

Chen and Wee [CW14a] introduce the notion of *dual system groups*, which is a notion that is inspired by the dual system encryption methodology. It defines a triple of groups with a non-degenerate bilinear mapping, accompanied by six algorithms that are defined over the groups such that certain properties hold. Then, these groups and their algorithms can be used to create encryption schemes in a generic fashion.

Attrapadung introduces the complementary notion of pair encoding in [Att14], which is another generic framework that can be used to analyze the security of encryption schemes, but rather considers the exponents of the group elements and how they relate to one another.

Subsequently, the work of Agrawal and Chase [AC16] combines the two notions in an explicit fashion, and generically constructs encryption schemes by using both of

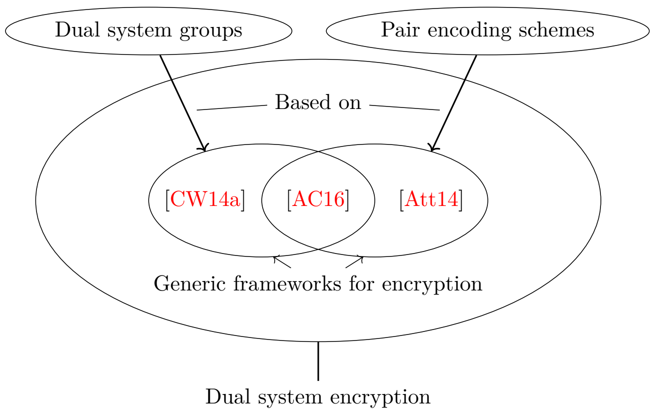


FIGURE A.1: A schematic overview of dual system encryption and generic frameworks based on dual system groups and pair encoding

them in a black-box manner, i.e. any definition of dual system groups that satisfies certain properties and any pair encoding scheme that is proven to be secure can be used to create a secure encryption scheme in a generic way. In the same work, they give a pair encoding scheme for the CP-ABE scheme with constant-size ciphertexts and prove that it is secure.

So in short, these works (i.e. [Att14; CW14a; AC16]) are all generic frameworks based on dual system encryption by using security notions of other primitives (see Fig. A.1 for a schematic overview of how they relate). The idea is that proving security for the other primitives is much easier than proving security of encryption schemes, and by proving that the construction of an encryption scheme from the primitive is secure, we can generically define secure schemes without having to prove each one of them fully secure conform Definition 18.

Before we will consider formal definitions of any of these notions, we will give a little bit more background information on each of the notions, i.e. dual system encryption, dual system groups and pair encoding.

A.1.1 Dual system encryption

	Normal CT	**S-F CT**
Normal SK	✓	✓
S-F SK	✓	✗

FIGURE A.2: Normal versus semi-functional keys and ciphertexts

In 2009, Waters [Wat09] presented a methodology called dual system encryption, which provides us with a method of devising encryption schemes that can be proven fully secure in the standard model under simple and established assumptions such as the decisional bilinear Diffie-Hellman (DBDH) and the decisional linear (DLIN) assumptions. In dual system encryption, there are two types of keys and ciphertexts, namely *normal* and *semi-functional* keys and ciphertexts. The normal keys and ciphertexts are the 'usual' keys and ciphertexts that are used in the encryption system. The semi-functional keys and ciphertexts are not used in the scheme itself, but they are used in the security proofs. Here, normal keys can decrypt semi-functional ciphertexts, and for the semi-functional keys holds that they can decrypt normal ciphertexts, but they cannot decrypt semi-functional ciphertexts.

The idea is that these semi-functional keys and ciphertexts are indistinguishable (either computationally, statistically or perfectly) from their normal counterparts. Then for the security proofs, we can define a series of games, which we will prove to be indistinguishable from one another. The first security game will be the real security game, in which the keys and ciphertexts are all normal. Then we define a series of security games in which we will substitute the normal ciphertexts for semi-functional ciphertexts, and because the normal and semi-functional ciphertexts are supposed to be indistinguishable, the attacker will not notice the difference between the first and each subsequent security game. Gradually, we will define a series of games that are indistinguishable from one another, replacing each normal key for a semi-functional key, such that the last game only distributes semi-functional keys and ciphertexts. Because semi-functional keys cannot decrypt semi-functional ciphertexts, an attacker can trivially not decrypt any challenge ciphertexts, and therefore the system will prove itself secure.

A.1.2 Dual system groups

In 2014, Chen and Wee [CW14a] introduced the notion of dual system groups, which leans on dual system encryption in the sense that the groups are defined in such a manner that we can generically prove security within the dual system encryption methodology. Chen and Wee identify the minimum number of properties that are necessary to apply dual system encryption techniques in the most basic setting of identity-based encryption, but they will also prove useful in other types of encryption such as attribute-based encryption.

These properties are defined over six randomized algorithms that are defined on the triple of groups $\mathbb{G}, \mathbb{H}, \mathbb{G}_T$. These algorithms are called 'sampling algorithms', such that for each sampling algorithm, we sample elements from $\mathbb{G}, \mathbb{H}$ or $\mathbb{G}_T$ in some random fashion. This 'random fashion' does not just simply mean that we are picking random elements from these groups, though. There are certain properties that must hold in order to prove correctness and security, which is why there are correctness and security properties on these sampling algorithms as well. For instance, samplings from $\mathbb{G}$ and $\mathbb{H}$ have to correspond in a special way, and the samplings have to be 'random enough' to be considered secure.

Like in dual system encryption, dual system groups distinguish between normal and semi-functional keys and ciphertexts, which is why two of these sampling algorithms generate elements in $\mathbb{G}$ and $\mathbb{H}$ respectively, such that they correspond with normal ciphertexts and keys. Two of the other sampling algorithms will generate semi-functional components in $\mathbb{G}$ and $\mathbb{H}$ such that we can create semi-functional ciphertexts and keys. Because we want to prove that normal and semi-functional keys are indistinguishable, as well as the normal and semi-functional ciphertexts, we need to define these sampling algorithms in such a fashion that this holds. One of the two remaining sampling algorithms generates $\mathbb{G}, \mathbb{H}$ and $\mathbb{G}_T$, and some other public and secret parameters, such as non-degenerate bilinear mapping $e : \mathbb{G} \times \mathbb{H} \to \mathbb{G}_T$, and the other sampling algorithm samples from $\mathbb{G}_T$ such that it corresponds with the first entry of the output of the sampling from $\mathbb{G}$.

A.1.3 Small introduction to pair encoding

In [Att14], Attrapadung introduces the notion of pair encoding as a major component in a framework that uses dual system encryption techniques, much like dual system groups. Whereas the notion of dual system groups focuses on the groups and how we can sample elements from those groups, pair encoding focuses on what happens in the exponents of the group elements in the key generation, encryption and decryption algorithms.

In this setting, what happens in the exponent of ciphertexts during encryption is called the *ciphertext encoding*, and analogously we introduce the term *key encoding* for the secret keys that are generated in the key generation algorithm. Decryption combines secret keys and ciphertexts in such a fashion that the plaintext is recovered. We will call the encoding of this 'combination' the *pair encoding*.

The idea is that the security of encryption schemes can be derived from the security of pair encoding schemes. Attrapadung formalizes the security properties of those pair encoding schemes in two notions: *perfectly master-key hiding* and *(co-)selectively master-key hiding*, from which we can construct, respectively, fully and selectively secure encryption schemes.

Agrawal and Chase [AC16] take this even further by finding a way to prove full security from a more relaxed version of perfect master-key hiding, which they coined as *relaxed perfect security*, which does not require perfect indistinguishability between distributions, but statistical indistinguishability. Moreover, they allow the addition of some extra noise to the key encoding, as long as adding this extra noise yields key encodings that are statistically close enough to the original key encoding. In order to know whether a pair encoding scheme is secure, we need to compute the statistical distance between two distributions (and either show that it is zero or negligible in the security parameter), which is considerably less complicated than proving security of an ABE scheme, and those proofs are typically much shorter than security proofs as well.

A.1.4 Intuitive notion of pair encoding

Now that we have established that it would be beneficial to consider the notion of pair encodings, we can elaborate on the specific encodings and how they relate.

So first, we consider groups $\mathbb{G}$ and $\mathbb{H}$ of order N with generators g and h such that we can define a non-degenerate bilinear mapping $e : \mathbb{G} \times \mathbb{H} \to \mathbb{G}_T$ to target group $\mathbb{G}_T$. As we have seen, it is often the case that part of the ciphertext is of the form $M \cdot e(g,h)^{\alpha s} \in \mathbb{G}_T$, where α is some secret that is generated and stored by the key generation authority, which we also refer to as the *master-key*, and s is the random contribution of the encryptor.

The idea is that we let (the other part of) the ciphertexts be defined in $\mathbb{G}$ and the secret keys in $\mathbb{H}$, so we can write the ciphertexts as vectors of the form $g^{\mathbf{c}(\mathbf{s},\mathbf{b})} = (g^{c_1(\mathbf{s},\mathbf{b})}, ..., g^{c_{w_1}(\mathbf{s},\mathbf{b})})$ and the keys in a similar fashion as vectors of the form $h^{\mathbf{k}(\alpha,\mathbf{r},\mathbf{b})}$, where $\mathbf{c}(\mathbf{s},\mathbf{b})$ denotes the ciphertext encoding and $\mathbf{k}(\alpha,\mathbf{r},\mathbf{b})$ denotes the key encoding. Note that we have introduced a couple of variables that we have not discussed yet. In the key encoding, we have input element α and vectors $\mathbf{r}$ and $\mathbf{b}$, and the inputs for ciphertext encoding are vectors $\mathbf{s}$ and $\mathbf{b}$, such that all of the entries of these variables are distributed uniformly over $\mathbb{Z}_N$. As the symbols already suggest, the key and ciphertext encoding have a common input, namely vector $\mathbf{b}$. So logically, they are also referred to as the *common variables*. In the key encoding, we use the master-key α and common variables $\mathbf{b}$, but we also include some key-specific random variables, i.e. $\mathbf{r} = (r_1, r_2, ...)$, which are used to tie the keys to one user and therefore avoid collusion, much like in the schemes we have seen before. In the ciphertext encoding, we have ciphertext-specific random variables $\mathbf{s} = (s, s_1, ...)$, where the first entry corresponds with the same s that is also used in ciphertext entry $M \cdot e(g,h)^{\alpha s}$, and the rest of the entries are to link the rest of the ciphertext entries together in a random fashion, also much like we have seen in other schemes. So we can express the ciphertext, with plaintext $M \in \mathbb{G}_T$, as

$$\mathbf{CT} = g^{\mathbf{c}(\mathbf{s},\mathbf{b})}, \qquad \mathrm{CT}_0 = M \cdot e(g,h)^{\alpha s} \tag{A.1}$$

and the secret keys as

$$\mathbf{SK} = h^{\mathbf{k}(\alpha,\mathbf{r},\mathbf{b})}, \tag{A.2}$$

where **CT** and **SK** both denote vectors (because **c** and **k** are also vectors).

To decrypt, we want to combine **CT** and **SK** in such a fashion that we somehow obtain $e(g,h)^{\alpha s}$, because then we can retrieve M from CT_0 by dividing by $e(g,h)^{\alpha s}$. Instead of looking at **CT** and **SK** directly, we can also consider **c** and **k** and how to combine them in order to obtain αs. We know that **c** is a function with input s, and **k** a function with input α, so we will look at each entry of **c** and each entry of **k**, and consider each pair of entries individually. If we define c_i as the i-th entry of **c** and k_j as the j-th entry of **k**, we can define for each i and j some $E_{i,j}$ such that

$$\sum_{i,j} E_{i,j} c_i k_j = \alpha s. \tag{A.3}$$

Then we can write **E** as a matrix with entries $E_{i,j}$, which denotes the pair encoding on the ciphertext and key encodings **c** and **k**. In terms of the ciphertext **CT** and keys **SK** this translates to computing

$$e(\mathbf{CT}^{\mathbf{E}}, \mathbf{SK}) = e(g^{\mathbf{c}(\mathbf{s},\mathbf{b})\mathbf{E}}, h^{\mathbf{k}(\alpha,\mathbf{r},\mathbf{b})}) = e(g,h)^{\sum_{i,j} E_{i,j} c_i k_j} = e(g,h)^{\alpha s}.$$

To use this in the CP-ABE setting, we define **k** over a set of attributes $\mathcal{S}$, and **c** over an access structure $\mathbb{A}$, such that applying the pair encoding **E** on the key and ciphertext encodings is only supposed to yield αs if $\mathbb{A} \models \mathcal{S}$.

Hence, whereas we had already discussed a correctness property in Eq. A.3, we can also define a security property for pair encoding schemes. Recall that, as per the security definition, if our set of attributes does not satisfy the access structure, then the ciphertext may not leak any information on M. To this end, we require α to be information-theoretically hidden, such that an attacker cannot observe the secret keys associated with different sets of attributes and learn something about α. In [Att14] and [AC16], Attrapadung, Agrawal and Chase attempt to avoid this type of attack by requiring that the distribution of the key and ciphertext encodings is independent on the choice of master-key α, i.e. the distribution of $\{\mathbf{c}(\mathbf{s},\mathbf{b}), \mathbf{k}(\alpha,\mathbf{r},\mathbf{b}) : \mathbf{s} = (s, s_1, ...), \mathbf{r} = (r_1, ...), \mathbf{b} = (b_1, ...),$ where $s, s_1, r_1, b_1, ... \in_R \mathbb{Z}_N\}$ is identical for each α[1]. This way, the same group elements will be distributed during the key generation and encryption algorithms, each with the same probability for any α, and therefore no attacker can guess α (or g^α) with non-negligible probability by observing a polynomial amount of keys. By extension, no information is revealed regarding plaintext M either, because $e(g,h)^{\alpha s}$ cannot be retrieved.

Implicitly, we have defined three algorithms already: the key encoding, the ciphertext encoding and the pair encoding, which are denoted as EncK, EncC and Pair respectively. In order to complete the definition of pair encoding schemes, we need a fourth algorithm that behaves like a setup, which we call Param, that defines some of the parameters such as the common variables $\mathbf{b} = (b_1, ..., b_n)$. Note that they are all distributed uniformly at random, i.e. $b_1, ..., b_n \in_R \mathbb{Z}_N$. We observe that in [Att14]

[1]Sometimes, statistical or even computational indistinguishability is sufficient, but depending on the generic construction, this yields weaker notions of security in the encryption scheme that follows from the encodings.

and [AC16], it is implied that the master-key α is generated in the key encoding EncK, whereas it is also strongly implied that this α is the same for each key that is generated after the Param algorithm is run. Moreover, this is also the case in other schemes, as we typically define the master secret key during the setup. This is why we include α in our parameter setup, and for α also holds that it is chosen uniformly, i.e. $\alpha \in_R \mathbb{Z}_N$.

In other words, a pair encoding scheme can be defined in terms of three randomized algorithms, namely Param, EncK and EncC, and algorithm Pair that depends on the outputs of EncK and EncC. In key encoding EncK, the key-specific entries of $\mathbf{r}$ are randomly distributed over $\mathbb{Z}_N$, and in ciphertext encoding EncC, the ciphtertext-specific entries of $\mathbf{s}$ are randomly distributed over $\mathbb{Z}_N$.

A.1.5 From pair encoding to attribute-based encryption

From our intuitive notion of pair encoding schemes, it becomes almost entirely clear how it relates to attribute-based encryption, as the algorithms of a pair encoding scheme are almost one-to-one related to the algorithms of attribute-based encryption. However, in [AC16], the ABE scheme is not entirely constructed as Equations A.1 and A.2 suggest. Instead, it combines the notions of dual system groups and pair encoding schemes, such that we can exploit the security properties of both notions.

If we use pair encodings in the sense that we described earlier, i.e. conform Eq. A.1 and A.2, we only know how to construct fully secure encryption schemes if the distribution of the ciphertext and key encodings is identical for each master-key α. If we use them in combination with dual system groups, we can also exploit the security properties that hold for these groups and the algorithms defined on them. As it turns out, this allows for a somewhat weaker notion of security in the pair encodings. Instead of requiring perfect indistinguishability, we only need statistical indistinguishability, and we also allow the addition of some noise, while the generic construction still results in a provably fully secure encryption scheme.

Recall that for dual system groups, we had defined six algorithms, of which one is a sampling algorithm that defines the groups $\mathbb{G}, \mathbb{H}, \mathbb{G}_T$ and other parameters, two are sampling algorithms for $\mathbb{G}$, two are sampling algorithms for $\mathbb{H}$ and one is a sampling algorithm for $\mathbb{G}_T$ that is somehow related to the first sampling algorithm for $\mathbb{G}$.

The idea is that we write the part of the ciphertext that was expressed as $g^{\mathbf{c}(\mathbf{s},\mathbf{b})}$ in terms of samplings from $\mathbb{G}$. Now, each entry of $g^{\mathbf{c}(\mathbf{s},\mathbf{b})}$ can be written in terms of generator g, ciphertext-specific randoms $\mathbf{s} = (s, s_1, ..., s_{w_2})$ and common variables $\mathbf{b} = (b_1, ..., b_n)$, as well as some constant coefficients that are defined in the vector $\mathbf{c}(\mathbf{s}, \mathbf{b})$. This means that we can write each entry $g^{c_\ell(\mathbf{s},\mathbf{b})}$ in terms of $g^{c_{\ell,i}(\mathbf{s})b_i}$ for all common variables $b_1, ..., b_n$, where $c_{\ell,i}(\mathbf{s}) = c_{\ell,i}(s, s_1, ..., s_{w_2})$ denotes the 'sub-polynomials' of $c_\ell(\mathbf{s}, \mathbf{b})$ such that $c_\ell(\mathbf{s}, \mathbf{b}) = \sum_{i=1}^{n} c_{\ell,i}(\mathbf{s})b_i$.

More specifically, this means that each ciphertext entry can be written as a product of 'generators' g^{b_i} with constants and ciphertext-specific random variables $c_{\ell,i}(\mathbf{s})$ in the exponent, i.e. $(g^{b_i})^{c_{\ell,i}(\mathbf{s})}$. In order to get our new formulation, we map all of the 'generators' g^{b_i} to entries of the outputs of the sampling algorithm defined on $\mathbb{G}$, and we let each variable in the ciphertext-specific randoms $s, s_1, ..., s_{w_2}$ correspond with the random value that is used in a sampling (so in total, we get $w_2 + 1$ samplings). The constants that were used in the old expression will remain the same under the mapping such that the structure of the encoding is preserved.

Note that the mapping of the original expression of the key encoding to our new expression of the key encodings works analogously to that of the ciphertext encoding, which we will see later this chapter.

For the other part of the ciphertext, which we formerly wrote as $M \cdot e(g,h)^{\alpha s}$, we use the sampling algorithm for group $\mathbb{G}_T$, which we denote as SampGT. This algorithm is related to the first entry of the outputs of the sampling function of $\mathbb{G}$ in some kind of fashion: the idea is that the public key is of the form $e(g_0, h^\alpha)$, where $g_0 = g^b$ for some $b \in \mathbb{Z}_N$ corresponds with the first entry the output of the sampling algorithm over $\mathbb{G}$. Then SampGT takes $e(g_0, h^\alpha)$ as input and outputs a randomized variant of it. More specifically, we use the same random value as in the first sampling from $\mathbb{G}$, i.e. s, so we get $e(g_0, h^\alpha)^s$ for $s \in_R \mathbb{Z}_N$. Then, the first entry of the output of the first sampling from $\mathbb{G}$ is similarly formed, i.e. g_0^s, and from the correctness properties of dual system groups and pair encoding schemes, follows that we can retrieve $e(g,h)^{\alpha b s}$ and finally also plaintext M.

A.2 Formal definitions

After giving some intuitive idea of dual system groups and pair encoding schemes, we can proceed to give formal definitions of both. For the dual system groups, we will use the definition that was formulated in [AC16], which is a more generalized version from the original definition in [CW14a]. For pair encoding schemes, we also use the definition that was given in [AC16], but we adapt it such that the generation of α is included in the Param algorithm, and not the EncK algorithm. We also use a narrower definition of pair encoding schemes than the original, which is defined over any predicate. Instead, we will only use 'CP-ABE predicates', i.e. if $\mathbb{A}$ is an access structure and $\mathcal{S}$ is a set of attributes, then the 'CP-ABE predicate' is true if and only if $\mathbb{A} \models \mathcal{S}$ holds.

Then, finally, we will give a generic construction of CP-ABE based on dual system groups and pair encoding schemes. This construction is based on the one that was given in [AC16], but the original version can construct any predicate encryption scheme from PES and DSG. Because predicate encryption is a generalization of attribute-based encryption, we can simply adapt the original construction to a CP-ABE setting.

A.2.1 Dual system groups

A dual system group is parameterized by the security parameter λ and a number n, which will correspond with the number of common variables in the pair encoding scheme.

Definition 21 (Dual System Groups (DSG) [AC16]) *A dual system group consists of six* PPT *algorithms, which we describe below.*

(i) SampP$(1^\lambda, 1^n)$: *On input 1^λ and 1^n,* SampP *outputs public parameters* PP *and secret parameters* SP*, which have the following properties:*

- PP *contains a triple of groups $(\mathbb{G}, \mathbb{H}, \mathbb{G}_T)$ and a non-degenerate, bilinear map $e : \mathbb{G} \times \mathbb{H} \to \mathbb{G}_T$, a homomorphism $\mu : \mathbb{H} \to \mathbb{G}_T$, and some additional parameters used by* SampG *and* SampH*, to be defined below. Given* PP*, we know the order of group $\mathbb{H}$ and how to sample uniformly from it. Let $N = \mathrm{ord}(\mathbb{H})$, such that N is a product of distinct primes of at least λ bits.*
- SP *contains a generator $\tilde{h} \in \mathbb{H}$ such that $\tilde{h} \neq 1_{\mathbb{H}}$.*

(ii) SampGT *takes an element in the image of μ and outputs another element from $\mathbb{G}_T$.*

(iii) SampG *and* SampH *take* PP *as input and output a vector of* $n+1$ *elements from* $\mathbb{G}$ *and* $\mathbb{H}$ *respectively.*

(iv) $\overline{\text{SampG}}$ *and* $\overline{\text{SampH}}$ *take* PP *and* SP *as input and output a vector of* $n+1$ *elements from* $\mathbb{G}$ *and* $\mathbb{H}$ *respectively.*

For these algorithms, we require the following properties to hold for all PP and SP that can be output by SampP. We define SampG_0 to be the first output entry of SampG. Analogously, we define SampH_0, $\overline{\text{SampG}}_0$ and $\overline{\text{SampH}}_0$.

Also note that all of these algorithms are randomized. Sometimes, however, we need to require that two algorithms use the same random integer, which we indicate by feeding the algorithm an extra input: we denote this extra input as σ, and we apply it to the algorithm by writing e.g. $\text{SampG}(\text{PP};\sigma)$, which means that SampG is applied to input PP with random variable σ.

First, we give a definition of correctness, and then we give a definition of security for dual system groups.

Definition 22 (DSG-correctness [AC16]) *A dual system group is correct if it satisfies the following two properties:*

(i) ***Projectivity:*** *For all* $h \in \mathbb{H}$ *and random integers* σ, $\text{SampGT}(\mu(h);\sigma) = e(\text{SampG}_0(\text{PP};\sigma), h)$.

(ii) ***Associativity:*** *If* $(g_0, ..., g_n)$ *and* $(h_0, ..., h_n)$ *are samples from* $\text{SampG}(\text{PP})$ *and* $\text{SampH}(\text{PP})$ *respectively, then for all* $i \in [1, n]$ *we have* $e(g_0, h_i) = e(g_i, h_0)$.

Note that, even though SampG and SampH sample from $\mathbb{G}$ and $\mathbb{H}$ in a random fashion, they still preserve a certain structure that the associativity property captures. We need this structure in order to write each element in the decryption that is of the form $e(g_i, h_0)^{\cdots}$ as $e(g_0, h_i)^{\cdots}$ such that, eventually, we can use the projectivity property to obtain $e(g_0, h)^{\alpha s}$ and retrieve the plaintext.

Definition 23 (DSG-security [AC16]) *A dual system group is secure if it satisfies the following three properties:*

(i) ***Orthogonality:*** *Generator* $\tilde{h}$ *in* SP *is in the kernel of* μ, *i.e.* $\mu(\tilde{h}) = 1_{\mathbb{G}_T}$.

(ii) ***Non-degeneracy:***

1. $\overline{\text{SampH}}_0(\text{PP},\text{SP}) \cong \tilde{h}^{\delta}$, *where* $\delta \in_R \mathbb{Z}_N$. *(Recall that* $\cong$ *indicates statistical indistinguishability.)*
2. *There is some generator* $\tilde{g} \in \mathbb{G}$ *such that* $\overline{\text{SampG}}_0(\text{PP},\text{SP}) \cong \tilde{g}^{\alpha}$, *where* $\alpha \in_R \mathbb{Z}_N$.
3. *For all* $g_0 \leftarrow \overline{\text{SampG}}_0(\text{PP},\text{SP})$, $e(g_0, \tilde{h})^{\beta}$ *is uniformly distributed over* $\mathbb{G}_T$, *where* $\beta \in_R \mathbb{Z}_N$.

(iii) ***Indistinguishability:*** *For two (positive) polynomials* $\mathrm{poly}_1(\cdot)$ *and* $\mathrm{poly}_2(\cdot)$*, define* $\mathbf{G}, \mathbf{H}, \hat{\mathbf{G}}, \hat{\mathbf{H}}, \hat{\mathbf{G}}'$ *and* $\hat{\mathbf{H}}'$ *as follows:*

$$\begin{aligned}
&(\mathrm{PP}, \mathrm{SP}) \leftarrow \mathrm{SampP}(1^\lambda, 1^n); \gamma_1, ..., \gamma_n \in_R \mathbb{Z}_N;\\
&\forall i \in [1, \mathrm{poly}_1(\lambda)] : \mathbf{g}_i := (g_{i,0}, ..., g_{i,n}) \leftarrow \mathrm{SampG}(\mathrm{PP});\\
&\qquad\qquad \hat{\mathbf{g}}_i := (\hat{g}_{i,0}, ..., \hat{g}_{i,n}) \leftarrow \overline{\mathrm{SampG}}(\mathrm{PP}, \mathrm{SP}); \hat{\mathbf{g}}'_i := (1, \hat{g}_{i,0}^{\gamma_1}, ..., \hat{g}_{i,0}^{\gamma_n});\\
&\forall j \in [1, \mathrm{poly}_2(\lambda)] : \mathbf{h}_j := (h_{j,0}, ..., h_{j,n}) \leftarrow \mathrm{SampH}(\mathrm{PP});\\
&\qquad\qquad \hat{\mathbf{h}}_j := (\hat{h}_{j,0}, ..., \hat{h}_{j,n}) \leftarrow \overline{\mathrm{SampH}}(\mathrm{PP}, \mathrm{SP}); \hat{\mathbf{h}}'_j := (1, \hat{h}_{j,0}^{\gamma_1}, ..., \hat{h}_{j,0}^{\gamma_n});\\
&\mathbf{G} := (\mathbf{g}_1, ..., \mathbf{g}_{\mathrm{poly}_1(\lambda)}); \hat{\mathbf{G}} := (\hat{\mathbf{g}}_1, ..., \hat{\mathbf{g}}_{\mathrm{poly}_1(\lambda)}); \hat{\mathbf{G}}' := (\hat{\mathbf{g}}'_1, ..., \hat{\mathbf{g}}'_{\mathrm{poly}_1(\lambda)});\\
&\mathbf{H} := (\mathbf{h}_1, ..., \mathbf{h}_{\mathrm{poly}_2(\lambda)}); \hat{\mathbf{H}} := (\hat{\mathbf{h}}_1, ..., \hat{\mathbf{h}}_{\mathrm{poly}_2(\lambda)}); \hat{\mathbf{H}}' := (\hat{\mathbf{h}}'_1, ..., \hat{\mathbf{h}}'_{\mathrm{poly}_2(\lambda)}).
\end{aligned}$$

We call a dual system group Left Subgroup Indistinguishable (LSI), Right Subgroup Indistinguishable (RSI) and Parameter Hiding (PH) if for all polynomials $\mathrm{poly}_1(\cdot)$ *and* $\mathrm{poly}_2(\cdot)$*, we have*

$$\{\mathrm{PP}, \mathbf{G}\} \approx \{\mathrm{PP}, \mathbf{G} \cdot \hat{\mathbf{G}}\} \tag{A.4}$$

$$\{\mathrm{PP}, \tilde{h}, \mathbf{G} \cdot \hat{\mathbf{G}}, \mathbf{H}\} \approx \{\mathrm{PP}, \tilde{h}, \mathbf{G} \cdot \hat{\mathbf{G}}, \mathbf{H} \cdot \hat{\mathbf{H}}\} \tag{A.5}$$

$$\{\mathrm{PP}, \tilde{h}, \hat{\mathbf{G}}, \hat{\mathbf{H}}\} \equiv \{\mathrm{PP}, \tilde{h}, \hat{\mathbf{G}} \cdot \hat{\mathbf{G}}', \hat{\mathbf{H}} \cdot \hat{\mathbf{H}}'\}. \tag{A.6}$$

Remarks: The security properties are all related to dual system encryption. For instance, we had already mentioned that the sampling functions are related to the normal keys and ciphertexts and semi-functional keys and ciphertexts. In this case, SampG and SampH distribute the normal ciphertexts and keys, and $\overline{\mathrm{SampG}}$ and $\overline{\mathrm{SampH}}$ distribute the semi-functional components of the ciphertexts and keys. In order to get a semi-functional ciphertext or key, we multiply some output of SampG or SampH with an output of $\overline{\mathrm{SampG}}$ and $\overline{\mathrm{SampH}}$ respectively. Because in dual system encryption, we require the step from a normal to a semi-functional ciphertext to be 'unnoticeable' to an attacker, the normal and semi-functional ciphertexts have to be indistinguishable, which is why the left subgroup indistinguishability, i.e. Eq. A.4, is important.

Something similar follows from the keys, however, in the following game we already have semi-functional ciphertexts, and we change the normal keys into semi-functional keys. This also has to go unnoticed by any attackers, which is why right subgroup indistinguishability is necessary, i.e. Eq. A.5. Then we have the parameter-hiding property in Eq. A.6, which is a little more difficult to explain intuitively. In practice, we cannot go from normal keys to semi-functional keys in one step, but first take a 'pit-stop' at keys that are 'in-between' normal and semi-functional. One of the reasons is that there is not enough randomness in the $n+1$ entries of the key: the entropy is only n, because of the same randomness that is used in $\mathrm{SampGT}(\mathrm{PP}; \sigma)$ as in the first entry of $\mathrm{SampG}(\mathrm{PP}; \sigma)$. The parameter-hiding property ensures that we can 'add' some randomness such that the entropy becomes $n+1$.

Then there is also the orthogonality property. Orthogonality is necessary to ensure that $e(\mathrm{SampG}_0(\mathrm{PP}; \sigma), \tilde{h}) = 1$. Because $e(\mathrm{SampG}_0(\mathrm{PP}; \sigma), \tilde{h}) = \mathrm{SampGT}(\mu(\tilde{h}); \sigma)$ holds with the projectivity property, and $\mathrm{SampGT}(\mu(\tilde{h}); \sigma) = \mathrm{SampGT}(1; \sigma)$ because $\mu(\tilde{h}) = 1$. Then from $\mu(1) = 1$ and $\mathrm{SampGT}(1; \sigma) = e(\mathrm{SampG}_0(\mathrm{PP}; \sigma), 1) = 1$ it follows that indeed $e(\mathrm{SampG}_0(\mathrm{PP}; \sigma), \tilde{h}) = 1$.

And finally, we have the non-degeneracy property, which is similar to the non-degeneracy property of bilinear mappings: it ensures that the sampling algorithms distribute all group elements with equal probability. Eventually, the non-degeneracy property ensures that the message in the semi-functional ciphertexts is information-theoretically hidden.

A.2.2 Pair encoding schemes

Now we consider a pair encoding scheme for any 'CP-ABE predicate', i.e. we have a universe of attributes $\mathcal{U}$ and a collection of access policies $\mathbb{A}$ over $\mathcal{U}$ such that for any set of attributes $\mathcal{S} \subseteq \mathcal{U}$ either $\mathbb{A} \models \mathcal{S}$ or $\mathbb{A} \not\models \mathcal{S}$ holds.

Note that, much like in the algorithms of the definition of dual system groups, we will use $\mathbb{Z}_N$, but not feed N as input to the algorithms like in [AC16]. We also use constants w_1 and w_2, which are used in the ciphertext encoding algorithm and denote the number of entries of the ciphertext encoding and the ciphertext-specific random vector $\mathbf{s}$ (minus 1) respectively. Similarly, m_1 and m_2 denote the number of entries in the key encoding and key-specific random vector $\mathbf{r}$ respectively. These are implicitly defined and together with the constants to be defined in the polynomials below (see Eq. A.7 and A.8), they are fixed for each invocation of any of the algorithms.

Definition 24 (Pair Encoding Schemes (PES) [AC16]) *For any 'CP-ABE predicate' and possibly some extra parameters* par *that are necessary to compute parameters $n, \mathbf{b}$ and α, a pair encoding scheme consists of four algorithms, which satisfy a correctness property that we define below.*

(i) Param(par) $\to (n, \mathbf{b}, \alpha)$*: The* Param *algorithm takes the extra parameters* par *as input, and outputs the number of common variables shared by the following two algorithms, which we denote with $n \in \mathbb{N}$, and we denote the common variables with $\mathbf{b} := (b_1, ..., b_n)$, which are picked uniformly at random from $\mathbb{Z}_N$. Moreover, the master-key $\alpha \in_R \mathbb{Z}$ is generated.*

(ii) EncC$(\mathbf{b}, \mathbb{A}) \to (\mathbf{c}(\mathbf{s}, \mathbf{b}) := (c_1, ..., c_{w_1}))$*: The* EncC *algorithm takes the common variables and access structure $\mathbb{A}$ as inputs and outputs a ciphertext encoding $c_1, ..., c_{w_1}$ which are entries that are expressed in terms of the entries of $\mathbf{s}$ and $\mathbf{b}$. In this case, $\mathbf{s} = (s, s_1, ..., s_{w_2})$ is randomly distributed over $\mathbb{Z}_N^{w_2+1}$, and more particularly, we can write each entry c_ℓ of the ciphertext encoding as a polynomial with constants $\zeta_\ell, \eta_{\ell,i}, \theta_{\ell,j}, \vartheta_{\ell,i,j} \in \mathbb{Z}_N$:*

$$c_\ell(\mathbf{s}, \mathbf{b}) := \zeta_\ell s + \sum_{i \in [1,w_2]} \eta_{\ell,i} s_i + \sum_{j \in [1,n]} \theta_{\ell,j} s b_j + \sum_{i \in [1,w_2], j \in [1,n]} \vartheta_{\ell,i,j} s_i b_j. \quad \text{(A.7)}$$

(iii) EncK$(\alpha, \mathbf{b}, \mathcal{S}) \to (\mathbf{k}(\alpha, \mathbf{r}, \mathbf{b}) := (k_1, ..., k_{m_1}))$*: The* EncK *algorithm takes the master-key, the common variables and set of attributes $\mathcal{S}$ as inputs and outputs a key encoding $k_1, ..., k_{m_1}$ which are entries that are expressed in terms of α and the entries of $\mathbf{r}$ and $\mathbf{b}$. In this case, $\mathbf{r} = (r_1, ..., r_{m_2})$ is randomly distributed over $\mathbb{Z}_N^{m_2}$ and more particularly, we can write each entry k_t of the key encoding as a polynomial with constants $\tau_t, \upsilon_{t,i'}, \phi_{t,i',j} \in \mathbb{Z}_N$:*

$$k_t(\alpha, \mathbf{r}, \mathbf{b}) := \tau_t \alpha + \sum_{i \in [1,m_2]} \upsilon_{t,i} r_i + \sum_{i \in [1,m_2], j \in [1,n]} \phi_{t,i,j} r_i b_j. \quad \text{(A.8)}$$

(iv) Pair$(\mathbb{A}, \mathcal{S}) \to \mathbf{E}$*: The* Pair *algorithm takes access structure $\mathbb{A}$ and set of attributes $\mathcal{S}$ as inputs and outputs a pair encoding, which is a $(m_1 \times w_1)$-matrix $\mathbf{E} \in \mathbb{Z}_N^{m_1 \times w_1}$.*

The Pair algorithm is supposed to yield a pair encoding $\mathbf{E}$ with entries $E_{t,\ell}$ such that for the ciphertext and key encoding with access structure $\mathbb{A}$ and set of attributes $\mathcal{S}$ holds that $\mathbb{A} \models \mathcal{S}$, the encodings can be combined in accordance with the pair encoding such that αs can be retrieved.

Definition 25 (PES-correctness [AC16]) *Let $\mathbb{A}$ be an access structure, and $\mathcal{S}$ a set of attributes. Let $\mathbf{c} \leftarrow \mathrm{EncC}(\mathbf{b}, \mathbb{A})$ and $\mathbf{k} \leftarrow \mathrm{EncK}(\alpha, \mathbf{b}, \mathbf{S})$ be a ciphertext and key encoding and $\mathbf{E} \leftarrow \mathrm{Pair}(\mathbb{A}, \mathcal{S})$ be the corresponding pair encoding. If $\mathbb{A} \models \mathcal{S}$, then we have*

$$\mathbf{k}\mathbf{E}\mathbf{c}^{\intercal} = \sum_{t\in[1,w_1],\ell\in[1,m_1]} k_t E_{t,\ell} c_\ell = \alpha s.$$

If $\mathbb{A} \not\models \mathcal{S}$, then $\mathbf{k}\mathbf{E}\mathbf{c}^{\intercal}$ outputs $\perp$.

Structural restriction: In [AC16], they impose a restriction on the pair encoding that [Att14] does not impose, but all of the pair encoding schemes that are formulated in [Att14] satisfy this restriction, so Agrawal and Chase argue that this does not make much of a difference. For the matrix we also require that $\mathbf{E}_{t,\ell}$ must be 0 if k_t and c_ℓ both have common variables in the expression.

Now we want to consider the security properties regarding those schemes. In [Att14], Attrapadung considers two types of security, perfect and computational, the perfect one being perfectly master-key hiding and the computational notions can be divided in two 'flavors', namely selectively and co-selectively secure master-key hiding. In [AC16], Agrawal and Chase only use Attrapadung's information-theoretic notion of security, i.e. perfectly master-key hiding, and introduce another information-theoretic notion of security called 'relaxed perfect security'. They only consider information-theoretic notions, because they lead to fully secure encryption schemes, contrary to the computational notions, which result in selectively secure encryption schemes.

Definition 26 (Perfect PES-Security [Att14; AC16]) *A pair encoding scheme* $(\mathrm{Param}, \mathrm{EncC}, \mathrm{EncK}, \mathrm{Pair})$ *is perfectly secure if for every order N, extra parameters* par*, access structure $\mathbb{A}$ and set of attributes $\mathcal{S}$ such that $\mathbb{A} \not\models \mathcal{S}$, we have that* $(\mathbf{c}(\mathbf{s}, \mathbf{b}), \mathbf{k}(0, \mathbf{r}, \mathbf{b}))$ *and* $(\mathbf{c}(\mathbf{s}, \mathbf{b}), \mathbf{k}(\alpha, \mathbf{r}, \mathbf{b}))$ *are perfectly indistinguishable for all* $\mathbf{s} \in_R \mathbb{Z}_N^{w_2+1}, \mathbf{b} \in_R \mathbb{Z}_N^n, \mathbf{r} \in_R \mathbb{Z}_N^{m_2}$ *and $\alpha \in_R \mathbb{Z}_N$.*

As we have already mentioned, Agrawal and Chase allow for some extra noise in the key encoding, as long as this noise is not 'too loud'. To this end, we define some extra sampling algorithm that samples this extra noise. More specifically, this sampling algorithm generates noise that will be added to the common variables in the key encoding, but not in the ciphertext encoding. The idea is that the noise will hide the master-key whilst avoiding that the attacker notices this. This way, in the security proofs, we can define a security game in which we generate noisy secret keys instead of the normal keys, and by design, the attacker will not notice this.

For each entry of $\mathbf{r}$, we will generate a new noisy vector, such that we can add the noise only to the common variables that are not 'canceled' when the rest of the entries of $\mathbf{r}$ are set to 0. Then if we add all partial noisy secret keys, we create a new, noisy secret key. If this noisy secret key is perfectly master-key hiding, then we can create a fully secure encryption scheme from it. But to this end, the sampling algorithm has to be independent on the choice of the access structure. If it is dependent, then the resulting encryption scheme is not fully secure, but semi-adaptively secure, which is a term that we used earlier in this work in Section 3.10.2. In this attack model, we need knowledge of the attacked access structure before the key queries are made, whereas in the full security model, the attacker is allowed to make key queries before announcing the attacked access structure.

Formally, we define $\mathbf{r}_d$ for all $d \in [1, m_2]$ to be the vector $\mathbf{r} = (r_1, ..., r_{m_2})$ with all entries set to 0, except for the d-th entry, i.e. $\mathbf{r}_d := (0, ..., 0, r_d, 0, ..., 0)$. The initial key encoding is $\mathbf{k}(\alpha, \mathbf{r}, \mathbf{b})$, and we define the partial key encoding as $\mathbf{k}(\alpha, \mathbf{r}_d, \mathbf{b})$, for which we sample some noise $\hat{\mathbf{b}}_d$, which we add to the common variables such that we get noisy partial key encoding $\mathbf{k}(\alpha, \mathbf{r}_d, \mathbf{b} + \hat{\mathbf{b}}_d)$. If we do this for each $d \in [1, m_2]$, we can make a noisy key by adding all partial keys, i.e.

$$\sum_{d \in [1, m_2]} \mathbf{k}(\alpha, \mathbf{r}_d, \mathbf{b} + \hat{\mathbf{b}}_d),$$

and of this noisy key, we will prove that it is master-key hiding, i.e. for all $\alpha \in \mathbb{Z}_N$ we have that distribution

$$\left\{ \left(\mathbf{c}(\mathbf{s}, \mathbf{b}), \sum_{d \in [1, m_2]} \mathbf{k}(\alpha, \mathbf{r}_d, \mathbf{b} + \hat{\mathbf{b}}_d) \right) : \mathbf{s}, \mathbf{b}, \mathbf{r} \in_R \mathbb{Z}_N^{n + w_2 + m_2 + 1}, \forall d : \hat{\mathbf{b}}_d \leftarrow \mathrm{Samp} \right\}$$

is statistically identical, which means that the statistical distance from distributions with $\alpha' \neq \alpha$ is negligible in the security parameter λ. Note that here we only require statistical instead of perfect indistinguishability, which is one of the two reasons that this security property is more relaxed than the initial security property.

Also observe that we have introduced the sampling algorithm Samp which is more specifically defined as:

Definition 27 (Noise sampling [AC16]) *We define the algorithm* $\mathrm{Samp}(d, \mathbb{A}, \mathcal{S})$ *as follows: for inputs* $d \in [1, m_2]$*, access structure* $\mathbb{A}$ *and set of attributes* $\mathcal{S}$*, it outputs a noise vector* $\hat{\mathbf{b}}_d = (\hat{b}_{d,1}, ..., \hat{b}_{d,n})$ *in* $\mathbb{Z}_N^n$*. Furthermore, we require that the probability that the algorithm yields* $(u \cdot \hat{b}_{d,1}, ..., u \cdot \hat{b}_{d,n})$ *is equal for each* $u \in \mathbb{Z}_N^*$*.*

Note that we do not require $\hat{\mathbf{b}}_d \in_R \mathbb{Z}_N^n$, but instead require that the algorithm yields $(u \cdot \hat{b}_{d,1}, ..., u \cdot \hat{b}_{d,n})$ with equal probability for each $u \in \mathbb{Z}_N^*$, because combining this property with the non-degeneracy and parameter-hiding DSG-security properties yields that adding this noise is unnoticeable by any attacker, which is essential in the security proof of the encryption scheme.

Now we can introduce the definition of relaxed security:

Definition 28 (Relaxed Perfect PES-Security [AC16]) *A pair encoding scheme is relaxed perfectly secure if there exists an algorithm* Samp *such that for every access structure* $\mathbb{A}$ *and set of attributes* $\mathcal{S}$ *such that* $\mathbb{A} \not\models \mathcal{S}$*, and every* $d \in [1, m_2]$ *we have:*

$$\{(\mathbf{c}(\mathbf{s}, \mathbf{b}), \mathbf{k}(0, \mathbf{r}_d, \mathbf{b}))\}_{\mathbf{s},\mathbf{r},\mathbf{b}} \cong \left\{ \left(\mathbf{c}(\mathbf{s}, \mathbf{b}), \mathbf{k}(0, \mathbf{r}_d, \mathbf{b} + \hat{\mathbf{b}}_d) \right) \right\}_{\mathbf{s},\mathbf{r},\mathbf{b},\hat{\mathbf{b}}_d \leftarrow \mathrm{Samp}(d, \mathbb{A}, \mathcal{S})}. \tag{A.9}$$

Furthermore,

$$\left\{ \left(\mathbf{c}(\mathbf{s}, \mathbf{b}), \sum_{d \in [1, m_2]} \mathbf{k}(0, \mathbf{r}_d, \mathbf{b} + \hat{\mathbf{b}}_d) \right) \right\} \cong \left\{ \left(\mathbf{c}(\mathbf{s}, \mathbf{b}), \sum_{d \in [1, m_2]} \mathbf{k}(\alpha, \mathbf{r}_d, \mathbf{b} + \hat{\mathbf{b}}_d) \right) \right\} \tag{A.10}$$

for all $\alpha \in \mathbb{Z}_N$*. Note that in Equations A.9 and A.10 the entries of* $\mathbf{s}, \mathbf{r}, \mathbf{b}$ *are distributed uniformly over* $\mathbb{Z}_N$ *and* $\hat{\mathbf{b}}_d \leftarrow \mathrm{Samp}(d, \mathbb{A}, \mathcal{S})$ *for all* $d \in [1, m_2]$*. Recall that*

$\cong$ *denotes statistical indistinguishability. Moreover, a pair encoding scheme satisfies strong relaxed perfect PES-security if* Samp *does not depend on* $\mathbb{A}$*.*

A.3 A generic CP-ABE construction from pair encoding

Previously, we have given intuitive explanations and formal definitions regarding the notions of dual system groups and pair encoding schemes, as well as their respective correctness and security properties. We will use them to give a generic construction of a CP-ABE scheme from any pair encoding scheme by using the notion of dual system groups. Note that we have adapted the generic construction of a predicate encryption scheme from a pair encoding scheme on predicates in [AC16] to a generic construction of a CP-ABE scheme.

A.3.1 Intuitive idea

Before we will introduce the actual construction, we will first give an intuitive idea as to how dual system groups and pair encoding are going to be combined in order to construct a CP-ABE scheme. On the one hand, we have the algorithms of pair encoding schemes that correspond with the algorithms of CP-ABE, i.e. the Setup and Param algorithms, the Encrypt and EncC algorithms, the KeyGen and EncK algorithm and the Decrypt and Pair algorithm.

On the other hand, we have dual system groups, which allow us to sample from $\mathbb{G}$ and $\mathbb{H}$ in a random fashion, such that certain correctness and security properties hold. These samples consist of $n+1$ group elements, which is one more than the number of common variables we have in our pair encoding scheme. The idea is that we let the sample algorithms of the dual system groups correspond with the variables that we have defined in our pair encoding scheme.

Recall that each ciphertext encoding entry is represented by a polynomial, i.e.

$$c_\ell(\mathbf{s},\mathbf{b}) := \zeta_\ell s + \sum_{i\in[1,w_2]} \eta_{\ell,i} s_i + \sum_{j\in[1,n]} \theta_{\ell,j} s b_j + \sum_{i\in[1,w_2],j\in[1,n]} \vartheta_{\ell,i,j} s_i b_j,$$

where $\mathbf{s} = (s, s_1, ..., s_{w_2})$ and $\mathbf{b} = (b_1, ..., b_n)$ denote the random and common variables respectively. So for each entry of $\mathbf{s}$, we can define a 'sub-polynomial' of c_ℓ, i.e.

$$c_\ell((s,0,...,0),\mathbf{b}) := s\left(\zeta_\ell + \sum_{j\in[1,n]} \theta_{\ell,j} b_j\right)$$
$$c_\ell((0,...,0,s_i,0,...,0),\mathbf{b}) := s_i\left(\eta_{\ell,i} + \sum_{j\in[1,n]} \vartheta_{\ell,i,j} b_j\right).$$

Observe how in all of these 'sub-polynomials' there occur exactly n common variables with constant coefficients and one constant. So if we let the constants correspond with the first entry of the output of SampG, and the n common variables with the other n entries, then we can let each 'sub-polynomial' correspond with one sampling from $\mathbb{G}$, i.e. for each $i \in [1, w_2+1]$ we sample $(g_{i,0}, ..., g_{i,n+1}) \leftarrow \text{SampG}(\text{PP})$. The randomness that comes from this algorithm will represent the random s or s_i in the ciphertext encoding, whereas the 'base' of $g_{i,j}$ represents the common variable b_j.

So if we combine the 'sub-polynomials' $c_{\ell,i}$ that we defined in Section A.1.5 and the 'sub-polynomials' that we defined above, i.e. let $\hat{c}_{\ell,i}(0) = c_{\ell,i}((s,0,...,0))/s$ and

$\hat{c}_{\ell,i}(j) = c_{\ell,i}((0,0,...,s_j,0,...,0))/s_i$, then we can define the mapping from the pair encoding and the dual system groups explicitly as

$$\mathrm{CT}_\ell = \prod_{i\in[0,w_2],j\in[1,n]} g_{i,j}^{\hat{c}_{\ell,i}(j)} = g_{0,0}^{\zeta_\ell} \cdot \prod_{i\in[1,w_2]} g_{i,0}^{\eta_{\ell,i}} \cdot \prod_{j\in[1,n]} g_{0,j}^{\theta_{\ell,j}} \cdot \prod_{i\in[1,w_2],j\in[1,n]} g_{i,j}^{\vartheta_{\ell,i,j}}.$$

We do something similar for the key encoding, i.e. we have $\mathbf{r} = (r_1,...,r_{m_2})$, so for each $j \in [1,m_2]$, we can sample $(h_{j,0},...,h_{j,n}) \leftarrow \mathrm{SampH(PP)}$, and we let each $h_{j,0}$ correspond with the constants and $h_{j,k}$ with the common variables. Note that the master-key α corresponds with master secret key MSK, which is uniformly distributed over $\mathbb{H}$. Then we can write the keys in a similar fashion as the ciphertexts.

Finally, let s be the random that was used in the first sampling of the ciphertext, i.e. $g_{0,0} = \tilde{g}^s$. Then with the projectivity property we have $e(\mathrm{SampG}_0(\mathrm{PP};s),\mathrm{MSK}) = \mathrm{SampGT}(\mu(\mathrm{MSK});s)$, and on the other hand we have $e(\mathrm{SampG}_0(\mathrm{PP};s),\mathrm{MSK}) = e(g_{0,0},\mathrm{MSK}) = e(\tilde{g},\mathrm{MSK})^s$, we can use $\mathrm{SampGT}(\mu(\mathrm{MSK});s)$ to hide message $M \in \mathbb{G}_T$ by simply setting $\mathrm{CT}_0 = M \cdot \mathrm{SampGT}(\mu(\mathrm{MSK});s)$. To this end, we assume that $\mu(\mathrm{MSK})$ is part of the public parameters.

A.3.2 The construction

The construction of a CP-ABE scheme from pair encoding and dual system groups follows relatively easy:

Construction 29 (Generic CP-ABE construction from PES) *A CP-ABE encryption scheme* $\Pi_{\mathrm{CP-ABE}} = (\mathrm{Setup},\mathrm{Encrypt},\mathrm{KeyGen},\mathrm{Decrypt})$ *for any attribute universe* $\mathcal{U}$ *and collection of access structures (i.e. the 'CP-ABE predicate') for which we have a pair encoding scheme* $\Gamma_{\mathrm{CP-ABE}} = (\mathrm{Param},\mathrm{EncC},\mathrm{EncK},\mathrm{Pair})$*, using dual system groups. The message space for* $\Pi_{\mathrm{CP-ABE}}$ *would be* $\mathbb{G}_T$*, generated by the* SampP *algorithm. Recall that* λ *denotes the security parameter,* $N \in \mathbb{N}$ *denotes the group order, which is the product of distinct primes of at least* λ *bits, and we have additional parameters* par.

(i) $\mathrm{Setup}(1^\lambda,\mathrm{par})$*: First run* Param(par) *to obtain* n *and the common variables* $\mathbf{b} = b_1,...,b_n$*, then run* $\mathrm{SampP}(1^\lambda,1^n)$ *to obtain* PP *and* SP*. Recall that given* PP*, we know the order of group* $\mathbb{H}$*, namely* $N = \mathrm{ord}(\mathbb{H})$*, and can sample uniformly from it. The output of the algorithm is*

$$\mathrm{MSK} \in_R \mathbb{H} \qquad \mathrm{MPK} := (\mathrm{PP},\mu(\mathrm{MSK})),$$

where MSK *denotes the secret key and* MPK *denotes the public key, which is published.*

Note that we have not explicitly used α *in this algorithm, but we could have easily defined our master secret key* MSK *as* h^α *for sampled* h*, i.e.* $h \leftarrow \mathrm{SampH}_0$.

(ii) $\mathrm{Encrypt}(\mathrm{MPK},\mathbb{A},M)$*: On input an access structure* $\mathbb{A}$*, and message* $M \in \mathbb{G}_T$*, run* $\mathrm{EncC}(\mathbf{b},\mathbb{A})$ *to obtain ciphertext encoding* $\mathbf{c}(\mathbf{s},\mathbf{b}) = (c_1,...,c_{w_1})$*, where* $\mathbf{s} = (s,s_1,...,s_{w_2})$ *is distributed uniformly. As per our intuitive description, draw* w_2+1 *samples:*

$$(g_{0,0},...,g_{0,n}) \leftarrow \mathrm{SampG}(\mathrm{PP};\sigma)$$

$$(g_{1,0},...,g_{1,n}) \leftarrow \mathrm{SampG}(\mathrm{PP}),...,(g_{w_2,0},...,g_{w_2,n}) \leftarrow \mathrm{SampG}(\mathrm{PP})$$

where σ *denotes the random that was used in drawing the first sample from* SampG*, and that we will also use in the last part of the ciphertext. Recall that* *the polynomial c_ℓ is given by*

$$c_\ell(\mathbf{s},\mathbf{b}) := \zeta_\ell s + \sum_{i\in[1,w_2]} \eta_{\ell,i} s_i + \sum_{j\in[1,n]} \theta_{\ell,j} s b_j + \sum_{i\in[1,w_2],j\in[1,n]} \vartheta_{\ell,i,j} s_i b_j.$$

Output $\mathrm{CT} := (\mathbb{A}, \mathrm{CT}_1, ..., \mathrm{CT}_{w_1}, \mathrm{CT}_{w_1+1})$ *as the encryption of M under $\mathbb{A}$, where*

$$\mathrm{CT}_\ell := g_{0,0}^{\zeta_\ell} \cdot \prod_{i\in[1,w_2]} g_{i,0}^{\eta_{\ell,i}} \cdot \prod_{j\in[1,n]} g_{0,j}^{\theta_{\ell,j}} \cdot \prod_{i\in[1,w_2],j\in[1,n]} g_{i,j}^{\vartheta_{\ell,i,j}}$$

for $\ell \in [1, w_1]$ and $\mathrm{CT}_{w_1+1} := M \cdot \mathrm{SampGT}(\mu(\mathrm{MSK}); \sigma)$*, where σ is the same random that was used in the first sampling.*

(iii) $\mathrm{KeyGen}(\mathrm{MSK}, \mathcal{S})$*: On input set of attributes $\mathcal{S}$, run* $\mathrm{EncK}(\alpha, \mathbf{b}, \mathcal{S})$ *to obtain key encoding $\mathbf{k}(\alpha,\mathbf{r},\mathbf{b}) = (k_1, ..., k_{m_1})$, where $\mathbf{r} = (r_1, ..., r_{m_2})$ is distributed uniformly. As per our intuitive description, draw m_2 samples:*

$$(h_{1,0}, ..., h_{1,n}) \leftarrow \mathrm{SampH}(\mathrm{PP}), ..., (h_{m_2,0}, ..., h_{m_2,n}) \leftarrow \mathrm{SampH}(\mathrm{PP}).$$

Recall that the polynomial k_t is given by

$$k_t(\alpha,\mathbf{r},\mathbf{b}) := \tau_t \alpha + \sum_{i\in[1,m_2]} \upsilon_{t,i} r_i + \sum_{i\in[1,m_2],j\in[1,n]} \phi_{t,i,j} r_i b_j.$$

Output the key as $\mathrm{SK} := (\mathcal{S}, \mathrm{SK}_1, ..., \mathrm{SK}_{m_1})$ *where for $t \in [1, m_1]$:*

$$\mathrm{SK}_t := \mathrm{MSK}^{\tau_t} \cdot \prod_{i\in[1,m_2]} h_{i,0}^{\upsilon_{t,i}} \cdot \prod_{i\in[1,m_2],j\in[1,n]} h_{i,j}^{\phi_{t,i,j}}.$$

(iv) $\mathrm{Decrypt}(\mathrm{MPK}, \mathrm{SK}, \mathrm{CT})$*: On input secret key* $\mathrm{SK} := (\mathcal{S}, \mathrm{SK}_1, ..., \mathrm{SK}_{m_1})$ *with set of attributes $\mathcal{S}$, and a ciphertext* $\mathrm{CT} := (\mathbb{A}, \mathrm{CT}_1, ..., \mathrm{CT}_{w_1+1})$ *with access structure $\mathbb{A}$, run* $\mathrm{Pair}(\mathbb{A}, \mathcal{S})$ *to obtain an $m_1 \times w_1$ matrix $\mathbf{E}$. Output*

$$\mathrm{CT}_{w_1+1} \cdot \left(\prod_{t\in[1,m_1],\ell\in[1,w_1]} e(\mathrm{CT}_\ell, \mathrm{SK}_t^{\mathbf{E}_{t,\ell}}) \right)^{-1}.$$

Correctness (sketch): Note that M is indeed recovered in the decryption algorithm, which follows from all of the correctness properties of dual system groups and pair encoding schemes:

Because of the structural restriction we have that only k_t or c_ℓ could have a common variable in the expression if $E_{t,\ell} \neq 0$. So for all t and ℓ such that $E_{t,\ell} \neq 0$, we either have that CT_ℓ cannot have $g_{i,j}^{\cdots}$ in the product, for any $j \neq 0$ or SK_t is cannot have $h_{i',j}^{\cdots}$ in the product for any $j \neq 0$. In the latter case, we can use that $e(g_{i,j}, h_{i',0}) = e(g_{i,0}, h_{i',j})$ holds (because of the associativity property), so we can write $e(\mathrm{CT}_\ell, \mathrm{SK}_t)$ in terms of $e(g_{i,0}, h_{i',j})$. Then multiplying for each t and ℓ and using the PES correctness property yields

$$\prod_{t\in[1,w_1],\ell\in[1,m_1]} e(\mathrm{CT}_\ell, \mathrm{SK}_t)^{E_{t,\ell}} = \prod_{\cdots} e(g_{i,0}, h_{i',j})^{\cdots} = e(\tilde{g}_0, \tilde{h})^{\alpha s} = e(\tilde{g}_0^s, \tilde{h}^\alpha),$$

for which $\tilde{g}_0$ denotes the 'base' of the first entry of the output of SampG, and $\tilde{h} \in \mathbb{H}$ is such that $\mathrm{MSK} = \tilde{h}^\alpha$. So we have obtained $e(g_{0,0}, \mathrm{MSK})$ which is equal to

$\mathrm{SampGT}(\mu(\mathrm{MSK}); s)$ because of the projectivity assumption, and therefore M can easily be retrieved.

A.3.3 Security of the construction

One of the reasons that we were considering the generic construction of encryption schemes from pair encoding schemes was that the security proofs of pair encoding schemes are much shorter than the security proofs of attribute-based encryption schemes. This does mean that we have to show that the encryption scheme that can be constructed from a pair encoding scheme with Construction 29 is secure if the pair encoding scheme is also secure. This leads to the following Theorem, which Agrawal and Chase prove in [AC16]:

Theorem 30 *If* $\Gamma_{\mathrm{CP-ABE}} = (\mathrm{Param}, \mathrm{EncC}, \mathrm{EncK}, \mathrm{Pair})$ *is a relaxed perfectly secure pair encoding scheme, then the CP-ABE scheme* $\Pi_{\mathrm{CP-ABE}} = (\mathrm{Setup}, \mathrm{Encrypt}, \mathrm{KeyGen}, \mathrm{Decrypt})$ *constructed in Construction 29 using* $\Gamma_{\mathrm{CP-ABE}}$ *is semi-adaptively secure. Furthermore, if the algorithm* Samp *does not depend on access structure* $\mathbb{A}$*, then* $\Pi_{\mathrm{CP-ABE}}$ *is fully secure.*

Proof sketch. This theorem is proven in [AC16]. We will not give the entire proof, which is roughly eight pages in LNCS style, but we can give a small, abstract outline of the proof structure. In their proof, they use a (very long) series of security games, which they prove to be statistically or computationally indistinguishable from one another by using the correctness and security properties of DSG and PES. The idea is that, this way, the first game (which represents the real security game) is proven to be computationally indistinguishable from the last (which is the security game with semi-functional ciphertexts and secret keys).

Whereas the first transition consists of the immediate step from normal ciphertexts to semi-functional ciphertexts, we cannot do the same for normal keys versus semi-functional keys, but instead use a series of intermediate games. Recall that sampling the normal keys could be done by using SampH, and the semi-functional components by using $\overline{\mathrm{SampH}}$.

First, we substitute the 'normal keys' for 'intermediate keys', which we do by using the noise sampling function to gradually add noise to all of the inputs to the key generation algorithm, until all of them are noisy. After this, all of the keys are 'pseudo-normal noisy keys'. The next step is to make the keys semi-functional noisy, which is done by making the master secret key semi-functional. Then, gradually, the noise is 'filtered out' by using the relaxed perfect security property as well as other DSG-properties that we have seen before, until all of the keys are semi-functional. Because the ciphertexts were semi-functional as well, we have reached the final game. Then the real security game is proven to be computationally indistinguishable from the last game, for which trivially holds that attackers cannot decrypt any challenge ciphertexts. □

Remark: Note that in the transition from a game with semi-functional ciphertexts and normal keys to the game with semi-functional ciphertexts and pseudo-normal noisy keys, we use the Samp algorithm to add noise to the keys. Because this is done before the key queries are made (i.e. the output noise is added to the input of the key generation algorithm) it matters whether Samp depends on access structure $\mathbb{A}$ or not. If so, then we obviously need to know the access structure that is going to be attacked before the key queries are made. This fits the semi-adaptive security model that we described in Section 3.10.2. If Samp does not need $\mathbb{A}$, then the CP-ABE scheme is fully secure.

Appendix B

A CP-ABE Scheme with Constant-Size Ciphertexts

In Appendix A, we have shown how we can generically construct a CP-ABE scheme from any pair encoding scheme, and how the security of the pair encoding scheme implies the security of the CP-ABE scheme that is constructed in this fashion. In this appendix, we will give an instantiation of a dual system group, which was introduced in [CW14a], and then we will give an instantiation of a pair encoding scheme, which was introduced in [AC16]. Then we will give the CP-ABE encryption scheme that we can construct with the generic construction in Construction 29.

B.1 A DSG-instantiation in prime-order groups

In [CW14a], Chen and Wee give an instantiation of a dual system group in a prime-order setting, which uses the SXDH assumption to prove right and left subgroup indistinguishability.

In Definition 21, we have shown that for the sampling algorithms, we require that several security properties hold, of which the subgroup indistinguishability properties are considerably the most difficult to integrate. Chen and Wee achieve these properties by using a slightly different group structure then we implied in Appendix A. Instead of simply using groups $\mathbb{G}$ and $\mathbb{H}$ of prime order p, where p is a prime of λ bits, and λ denotes the security parameter, they use groups $\mathbb{G} = G \times G$ and $\mathbb{H} = H \times H$ such that G and H are groups of order p. We write each element in $\mathbb{G}$ as $g^{\mathbf{x}}$, where $\mathbf{x} = (x_1, x_2) \in \mathbb{Z}_p^2$, and analogously for elements of $\mathbb{H}$. The non-degenerate bilinear mapping is defined over G and H, i.e. $e : G \times H \to G_T$.

Now, we let $\mathbf{B}$ be an invertible 2×2 matrix, that was distributed uniformly over $\mathbb{Z}_p^{2\times 2}$, and we define $\mathbf{B}^* := (\mathbf{B}^\intercal)^{-1}$ as the inverse of the transpose of $\mathbf{B}$.

We can use SXDH to ensure left and right subgroup indistinguishability. For this, we split $\mathbf{B}$ into two halves, i.e. the left part $\begin{pmatrix}\mathbf{B}_{11}\\ \mathbf{B}_{21}\end{pmatrix}$ and the right part $\begin{pmatrix}\mathbf{B}_{12}\\ \mathbf{B}_{22}\end{pmatrix}$. Now, if we use the left part in SampG_0, for which we use randomness $s \in_R \mathbb{Z}_p$, and the right part in $\overline{\mathrm{SampG}_0}$, for which we use randomness $\hat{s}$, then we can write those outputs as

$$\mathbf{g}_0 = (g^{s\mathbf{B}_{11}}, g^{s\mathbf{B}_{21}}) \qquad \hat{\mathbf{g}}_0 = (g^{\hat{s}\mathbf{B}_{12}}, g^{\hat{s}\mathbf{B}_{22}}).$$

Recall that left subgroup indistinguishability requires that $\mathbf{g}_0$ and $\mathbf{g}_0 \cdot \hat{\mathbf{g}}_0$ are computationally indistinguishable, for which $\hat{\mathbf{g}}_0$ is the output of $\overline{\mathrm{SampG}_0}$. So we have

$$\mathbf{g}_0 = g^{\mathbf{B}\begin{pmatrix}s\\0\end{pmatrix}}, \quad \mathbf{g}_0 \cdot \hat{\mathbf{g}}_0 = (g^{s\mathbf{B}_{11}+\hat{s}\mathbf{B}_{12}}, g^{s\mathbf{B}_{21}+\hat{s}\mathbf{B}_{22}}) = g^{\mathbf{B}\begin{pmatrix}s\\ \hat{s}\end{pmatrix}}.$$

Now, for $\tilde{\mathbf{g}}_0 = g^{\mathbf{B}\begin{pmatrix} s \\ \tilde{s} \end{pmatrix}}$, we have $\tilde{\mathbf{g}}_0 = \mathbf{g}_0$ if $\tilde{s} = 0$ and $\tilde{\mathbf{g}}_0 = \mathbf{g}_0 \cdot \hat{\mathbf{g}}_0$ if $\tilde{s} \in_R \mathbb{Z}_p^*$.

On the other hand, we have the SXDH assumption, which assumes that we cannot distinguish $g^{a_2 s_1}$ from $g^{a_2 s_1 + s_2}$, given g^{a_1}, g^{a_2} and $g^{a_1 s_1}$. Then if we can distinguish $\mathbf{g}_0$ from $\mathbf{g}_0 \cdot \hat{\mathbf{g}}_0$ with non-negligible advantage in polynomial time, then we can also break SXDH.

Let $g^{\mathbf{W}} = \begin{pmatrix} g^{a_1} & 1 \\ g^{a_2} & g \end{pmatrix}$, and $\tilde{\mathbf{B}}$ be a uniformly distributed invertible matrix. Then we can compute, implicitly, a uniformly distributed $\mathbf{B}$, i.e. $g^{\mathbf{B}} = g^{\tilde{\mathbf{B}}\mathbf{W}}$. Let

$$\tilde{\mathbf{g}}_0 = \begin{pmatrix} g^{a_1 s_1 \tilde{\mathbf{B}}_{11} + (a_2 s_1 + s_2)\tilde{\mathbf{B}}_{12}} \\ g^{a_1 s_1 \tilde{\mathbf{B}}_{21} + (a_2 s_1 + s_2)\tilde{\mathbf{B}}_{22}} \end{pmatrix} = g^{\tilde{\mathbf{B}}\begin{pmatrix} a_1 s_1 \\ a_2 s_1 + s_2 \end{pmatrix}} = g^{\mathbf{B}\begin{pmatrix} s_1 \\ s_2 \end{pmatrix}}.$$

Note that we can compute $\begin{pmatrix} g^{a_1 s_1 \tilde{\mathbf{B}}_{11} + (a_2 s_1 + s_2)\tilde{\mathbf{B}}_{12}} \\ g^{a_1 s_1 \tilde{\mathbf{B}}_{21} + (a_2 s_1 + s_2)\tilde{\mathbf{B}}_{22}} \end{pmatrix}$ because we have $g^{a_1 s_1}$ and $a_2 s_1 + s_2$ for which we have to determine whether $s_2 = 0$ or $s_2 \in \mathbb{Z}_p^*$.

Because we can distinguish between $\mathbf{g}_0$ and $\mathbf{g}_0 \cdot \hat{\mathbf{g}}_0$, we know whether $s_2 = 0$ or $s_2 \in_R \mathbb{Z}_p^*$, which also solves SXDH. Because SXDH is widely considered to be intractable, this is a contradiction and therefore it follows that we cannot distinguish $\mathbf{g}_0$ and $\mathbf{g}_0 \cdot \hat{\mathbf{g}}_0$ with non-negligible advantage.

Hence, using invertible matrix $\mathbf{B}$ seems like a good choice for the first output entry of SampG. For the other n entries of the output, we use a matrix $\mathbf{A}_i$ that is entirely uniformly distributed over $\mathbb{Z}_p$ for each entry. Then we will compute $\mathbf{B}\mathbf{A}_i$ for each $i \in [1, n]$, which will be the matrix that corresponds with the i-th output entry of SampG. In a familiar fashion, it also follows that for $\tilde{\mathbf{g}}_i = g^{\mathbf{B}\mathbf{A}_i\begin{pmatrix} s \\ \hat{s} \end{pmatrix}}$ we cannot distinguish whether $\hat{s} = 0$ or in $\mathbb{Z}_p^*$ under the SXDH assumption.

We will use this in order to define the sample algorithms. But before we can define our instantiation of the sample algorithms, we need give a couple of notations and definitions first. We define $\mathcal{G}$ to be a generator that takes as input a security parameter λ and outputs a description of (p, G, H, G_T, g, h, e), where p is a prime of at least λ bits, G, H and G_T are cyclic groups of order p, and g, h are generators of G and H respectively. Finally, $e : G \times H \to G_T$ is a non-degenerate bilinear mapping. We also define π_L and π_R, which denote two projection functions that map a 2×2-matrix to its left and right column respectively.

We also introduce some special vector operations in the exponent, i.e. $e(g^{\mathbf{x}}, h^{\mathbf{y}}) = e(g, h)^{\mathbf{x}^\intercal \mathbf{y}}$. Analogously, we have for matrices $e(g^{\mathbf{A}}, h^{\mathbf{B}}) = e(g, h)^{\mathbf{A}^\intercal \mathbf{B}}$. For given $g^{\mathbf{A}}$ and $h^{\mathbf{B}}$, we can compute $e(g, h)^{\mathbf{A}^\intercal \mathbf{B}}$ by computing $e(g^{\mathbf{A}_{k,i}}, h^{\mathbf{B}_{k,j}})$ for each $k \in [1, 2]$ and then multiply: $e(g, h)^{\mathbf{A}^\intercal \mathbf{B}} = e(g^{\mathbf{A}_{1,i}}, h^{\mathbf{B}_{1,j}}) \cdot e(g^{\mathbf{A}_{2,i}}, h^{\mathbf{B}_{2,j}})$.

B.1.1 Definitions of the sample algorithms

Now we will give the definitions of the sample algorithms.

(i) SampP$(1^\lambda, 1^n)$: Let $(p, G, H, \mathbb{G}_T, g, h, e) \leftarrow \mathcal{G}(\lambda)$, and define $(\mathbb{G}, \mathbb{H}, \mathbb{G}_T, e) := (G \times G, H \times H, \mathbb{G}_T, e)$. Pick invertible $\mathbf{B} \in_R \mathbb{Z}_p^{2\times 2}$, such that $\mathbf{B}^* := (\mathbf{B}^{-1})^\intercal$ is well-defined, and let $\mathbf{A}_1, ..., \mathbf{A}_n \in_R \mathbb{Z}_p^{2\times 2}$. Furthermore, we define diagonal matrix $\mathbf{R} \in_R \mathbb{Z}_p^{2\times 2}$ such that $\mathbf{R}_{11} \neq 0$ and $\mathbf{R}_{22} = 1$. Then we define the following matrices and vectors:

$$\begin{array}{llll} \mathbf{D} := \pi_L(\mathbf{B}), & \mathbf{f} := \pi_R(\mathbf{B}), & \mathbf{D}_i := \pi_L(\mathbf{B}\mathbf{A}_i), & \mathbf{f}_i := \pi_R(\mathbf{B}\mathbf{A}_i) \\ \mathbf{D}^* := \pi_L(\mathbf{B}^*\mathbf{R}), & \mathbf{f}^* := \pi_R(\mathbf{B}^*\mathbf{R}), & \mathbf{D}_i^* := \pi_L(\mathbf{B}^*\mathbf{A}_i^\intercal\mathbf{R}), & \mathbf{f}_i^* := \pi_R(\mathbf{B}^*\mathbf{A}_i^\intercal\mathbf{R}) \end{array}$$

Also define $\mu : \mathbb{H} \to \mathbb{G}_T$ by $\mu(h^{\mathbf{k}}) = e(g^{\mathbf{D}}, h^{\mathbf{k}})$ for all $\mathbf{k} \in \mathbb{Z}_p^2$ and set $\tilde{\mathbf{h}} := h^{\mathbf{f}^*}$. We define:

$$\begin{array}{c} \mathbf{g}_0 = g^{\mathbf{D}}, \mathbf{g}_1 = g^{\mathbf{D}_1}, ..., \mathbf{g}_n = g^{\mathbf{D}_n} \\ \mathbf{h}_0 = h^{\mathbf{D}^*}, \mathbf{h}_1 = h^{\mathbf{D}_1^*}, ..., \mathbf{h}_n = h^{\mathbf{D}_n^*} \end{array}.$$

Then the output consists of the public and secret parameters

$$\mathrm{PP} = \left((\mathbb{G}, \mathbb{H}, \mathbb{G}_T, e); \mu; \begin{array}{c} \mathbf{g}_0, \mathbf{g}_1, ..., \mathbf{g}_n \\ \mathbf{h}_0, \mathbf{h}_1, ..., \mathbf{h}_n \end{array} \right), \quad \mathrm{SP} = \left(\begin{array}{c} g^{\mathbf{f}}, g^{\mathbf{f}_1}, ..., g^{\mathbf{f}_n} \\ h^{\mathbf{f}^*}, h^{\mathbf{f}_1^*}, ..., h^{\mathbf{f}_n^*} \end{array} \right).$$

(ii) $\mathrm{SampGT}(g_T^p)$: Pick $s \in_R \mathbb{Z}_p$, and output $g_T^{ps} \in \mathbb{G}_T$.

(iii) $\mathrm{SampG}(\mathrm{PP})$: Pick $s \in_R \mathbb{Z}_p$, and output $(g^{\mathbf{D}s}, g^{\mathbf{D_1}s}, ..., g^{\mathbf{D_n}s}) \in \mathbb{G}^{n+1}$.

(iv) $\mathrm{SampH}(\mathrm{PP})$: Pick $r \in_R \mathbb{Z}_p$, and output $(h^{\mathbf{D}^*r}, h^{\mathbf{D}_1^*r}, ..., h^{\mathbf{D}_n^*r}) \in \mathbb{H}^{n+1}$.

(v) $\overline{\mathrm{SampG}}(\mathrm{PP}, \mathrm{SP})$: Pick $\hat{s} \in_R \mathbb{Z}_p^*$, and output $(g^{\mathbf{f}\hat{s}}, g^{\mathbf{f_1}\hat{s}}, ..., g^{\mathbf{f_n}\hat{s}}) \in \mathbb{G}^{n+1}$.

(vi) $\overline{\mathrm{SampH}}(\mathrm{PP}, \mathrm{SP})$: Pick $\hat{r} \in_R \mathbb{Z}_p^*$, and output $(h^{\mathbf{f}^*\hat{r}}, h^{\mathbf{f}_1^*\hat{r}}, ..., h^{\mathbf{f}_n^*\hat{r}}) \in \mathbb{H}^{n+1}$.

Note that the use of capital letters $\mathbf{D}$, etc. insinuates that they are matrices, but in fact, are vectors. The reason for this is that we have used the notations that are used in [CW14a], and in the more general sense, they denote matrices.

Lemma 31 *The dual system group that can be defined in terms of these algorithms satisfies the correctness properties in Definition 22 and the security properties in Definition 23.*

Proof sketch. In [CW14a], Chen and Wee prove this for a more general case. Whereas we will not give an extensive proof, we will give an intuition regarding the properties and why they hold for our sample functions.

The correctness properties hold because

(i) **Projectivity:** For all $h \in \mathbb{H}$ and random $\sigma \in \mathbb{Z}_p$ we have $\mathrm{SampGT}(\mu(h); \sigma) = \mu(h)^\sigma = e(g^{\mathbf{D}}, h)^\sigma = e(g^{\mathbf{D}\sigma}, h) = e(\mathrm{SampG}_0(\mathrm{PP}; \sigma), h)$.

(ii) **Associativity:** If $(\overline{g}_0, ..., \overline{g}_n) \leftarrow \mathrm{SampG}(\mathrm{PP})$ and $(\overline{h}_0, ..., \overline{h}_n) \leftarrow \mathrm{SampH}(\mathrm{PP})$, then for all $i \in [1, n]$ we have $e(\overline{g}_0, \overline{h}_i) = e(g^{\mathbf{D}s}, h^{\mathbf{D}_i^*r}) = e(g, h)^{\mathbf{D}^\intercal\mathbf{D}_i^*sr} = e(g, h)^{\pi_L(\mathbf{B})^\intercal\pi_L(\mathbf{B}^*\mathbf{A}_i^t\mathbf{R})sr} = e(g, h)^{\pi_L(\mathbf{A}_i^\intercal\mathbf{R})sr} = e(g, h)^{\pi_L(\mathbf{B}\mathbf{A}_i)^\intercal\pi_L(\mathbf{B}^*\mathbf{R})sr} = e(g, h)^{\mathbf{D}_i^\intercal\mathbf{D}^*sr} = e(g^{\mathbf{D}_is}, h^{\mathbf{D}^*r}) = e(\overline{g}_i, \overline{h}_0)$, where $s, r \in_R \mathbb{Z}_p$ the random integers that were used in the samplings.

The security properties are fairly straightforward: the non-degeneracy properties follow almost immediately from the definition of the sample functions, and we had already argued a case for left and right subgroup indistinguishability. The orthogonality property follows from $\mathbf{D}^\intercal\mathbf{f}^* = \pi_L(\mathbf{B})^\intercal\pi_R(\mathbf{B}^*\mathbf{R}) = 0$, i.e. $\mu(\tilde{\mathbf{h}}) = e(g^{\mathbf{D}}, h^{\mathbf{f}^*}) = e(g, h)^{\mathbf{D}^\intercal\mathbf{f}^*} = 1_{G_T}$.

For the parameter hiding property, we have to show that given PP and $\tilde{\mathbf{h}}$, we cannot easily distinguish $\hat{\mathbf{g}} = (\hat{g}_0, ..., \hat{g}_n) \leftarrow \overline{\mathrm{SampG}}(\mathrm{PP}, \mathrm{SP})$ from $\hat{\mathbf{g}}' = \hat{\mathbf{g}} \cdot (1, \hat{g}_0^{\gamma_1}, ..., \hat{g}_0^{\gamma_n})$, and $\hat{\mathbf{h}} = (\hat{h}_0, ..., \hat{h}_n) \leftarrow \overline{\mathrm{SampH}}(\mathrm{PP}, \mathrm{SP})$ from $\hat{\mathbf{h}}' = \hat{\mathbf{h}} \cdot (1, \hat{h}_0^{\gamma_1}, ..., \hat{h}_0^{\gamma_n})$, where $\gamma_i \in_R \mathbb{Z}_p$

randomly distributed. Let $\hat{s}$ and $\hat{r}$ be the randoms that are used in $\overline{\mathsf{SampG}}$ and $\overline{\mathsf{SampH}}$, then

$$\begin{aligned}\hat{\mathbf{g}} &= (g^{\mathbf{f}\hat{s}}, g^{\mathbf{f}_1\hat{s}}, ..., g^{\mathbf{f}_n\hat{s}}) \\ \hat{\mathbf{g}}' &= (g^{\mathbf{f}\hat{s}}, g^{\mathbf{f}_1\hat{s}+\gamma_1\mathbf{f}\hat{s}}, ..., g^{\mathbf{f}_n\hat{s}+\gamma_n\mathbf{f}\hat{s}})\end{aligned} \quad \text{and} \quad \begin{aligned}\hat{\mathbf{h}} &= (h^{\mathbf{f}^*\hat{r}}, h^{\mathbf{f}_1^*\hat{r}}, ..., h^{\mathbf{f}_n^*\hat{r}}) \\ \hat{\mathbf{h}}' &= (h^{\mathbf{f}^*\hat{r}}, h^{\mathbf{f}_1^*\hat{r}+\gamma_1\mathbf{f}^*\hat{r}}, ..., h^{\mathbf{f}_n^*\hat{r}+\gamma_n\mathbf{f}^*\hat{r}})\end{aligned}.$$

We have

$$\begin{aligned}\mathbf{f}_i\hat{s} &= \pi_R(\mathbf{B}\mathbf{A}_i)\hat{s}, & \mathbf{f}_i\hat{s} + \gamma_i\mathbf{f}\hat{s} &= \pi_R(\mathbf{B}\mathbf{A}_i)\hat{s} + \gamma_i\pi_R(\mathbf{B})\hat{s} \\ \mathbf{f}_i^*\hat{r} &= \pi_R(\mathbf{B}^*\mathbf{A}_i^\intercal\mathbf{R})\hat{r}, & \mathbf{f}_i^*\hat{r} + \gamma_i\mathbf{f}^*\hat{r} &= \pi_R(\mathbf{B}^*\mathbf{A}_i^\intercal\mathbf{R})\hat{r} + \gamma_i\pi_R(\mathbf{B}^*\mathbf{R})\hat{r},\end{aligned}$$

which we can also write as

$$\begin{aligned}\mathbf{f}_i\hat{s} &= \begin{pmatrix}((\mathbf{A}_i)_{12}\mathbf{B}_{11} + \boxed{(\mathbf{A}_i)_{22}}\mathbf{B}_{12})\hat{s} \\ ((\mathbf{A}_i)_{12}\mathbf{B}_{21} + \boxed{(\mathbf{A}_i)_{22}}\mathbf{B}_{22})\hat{s}\end{pmatrix} \\ \mathbf{f}_i\hat{s} + \gamma_i\mathbf{f}\hat{s} &= \begin{pmatrix}((\mathbf{A}_i)_{12}\mathbf{B}_{11} + \boxed{((\mathbf{A}_i)_{22} + \gamma_i)}\mathbf{B}_{12})\hat{s} \\ ((\mathbf{A}_i)_{12}\mathbf{B}_{21} + \boxed{((\mathbf{A}_i)_{22} + \gamma_i)}\mathbf{B}_{22})\hat{s}\end{pmatrix} \\ \mathbf{f}_i^*\hat{r} &= \frac{\hat{r}}{\mathbf{B}_{11}\mathbf{B}_{22} - \mathbf{B}_{12}\mathbf{B}_{21}}\begin{pmatrix}(\mathbf{A}_i)_{21}\mathbf{B}_{22} - \boxed{(\mathbf{A}_i)_{22}}\mathbf{B}_{21} \\ -(\mathbf{A}_i)_{21}\mathbf{B}_{12} + \boxed{(\mathbf{A}_i)_{22}}\mathbf{B}_{11}\end{pmatrix} \\ \mathbf{f}_i^*\hat{r} + \gamma_i\mathbf{f}^*\hat{r} &= \frac{\hat{r}}{\mathbf{B}_{11}\mathbf{B}_{22} - \mathbf{B}_{12}\mathbf{B}_{21}}\begin{pmatrix}(\mathbf{A}_i)_{21}\mathbf{B}_{22} - \boxed{((\mathbf{A}_i)_{22} + \gamma_i)}\mathbf{B}_{21} \\ -(\mathbf{A}_i)_{21}\mathbf{B}_{12} + \boxed{((\mathbf{A}_i)_{22} + \gamma_i)}\mathbf{B}_{11}\end{pmatrix}.\end{aligned}$$

Note that we have indicated the differences between $\mathbf{f}_i\hat{s}$ and $\mathbf{f}_i\hat{s} + \gamma_i\mathbf{f}\hat{s}$, as well as the differences between $\mathbf{f}_i^*\hat{r}$ and $\mathbf{f}_i^*\hat{r} + \gamma_i\mathbf{f}^*\hat{r}$ by putting $\boxed{\text{boxes}}$ around those specific parts in the equation. Because of these boxes, we can immediately see that the only difference between $(\mathbf{f}_i\hat{s}, \mathbf{f}_i^*\hat{r})$ and $(\mathbf{f}_i\hat{s} + \gamma_i\mathbf{f}\hat{s}, \mathbf{f}_i^*\hat{r} + \gamma_i\mathbf{f}^*\hat{r})$ is that $(\mathbf{A}_i)_{22}$ has some 'extra randomness' (i.e. γ_i) in the second expression. However, the distributions of $(\mathbf{A}_i)_{22}$ and $(\mathbf{A}_i)_{22} + \gamma_i$ for $(\mathbf{A}_i)_{22}, \gamma_i \in_R \mathbb{Z}_p$ are identical. Hence, the distributions of $(\hat{\mathbf{g}}, \hat{\mathbf{h}})$ and $(\hat{\mathbf{g}}', \hat{\mathbf{h}}')$ are identical. □

B.2 A PES-instantiation with short ciphertext encoding

The second instantiation that we need, and for this, we also assume that our groups have a prime order, is a pair encoding scheme that was introduced by Agrawal and Chase in [AC16]. The novelty of this scheme is that the ciphertext encoding consists of a constant number of elements, and therefore leads to the first provably semi-adaptively secure CP-ABE scheme with constant-size ciphertexts. We will give a description of the scheme, and then sketch the security proof of Agrawal and Chase.

B.2.1 The constant-size ciphertext encoding PES

Recall that an LSSS access structure is denoted as $\mathbb{A} = (A, \rho)$, where A is a $n_1 \times n_2$ matrix with entries in $\mathbb{Z}_p$, where p is a prime of at least λ bits, and $\rho : [1, n_1] \to \mathcal{U}$, which maps the rows of A to $\mathcal{U}$, where $\mathcal{U} = \mathbb{Z}_p$ denotes the (large) universe of attributes. We denote A_i as the i-th row of A and $a_{i,j} = A_{ij}$ as the j-th element of the i-th row. For set of attributes $\mathcal{S} \subseteq \mathcal{U}$ we define $\Upsilon = \{i \in [1, n_1] : \rho(i) \in \mathcal{S}\}$ to be the set of indices of rows in A associated with $\mathcal{S}$. Furthermore, for this scheme, we define $T \in \mathbb{N}$ to be the maximum number of attributes a user is allowed to possess, i.e. $|\mathcal{S}| \leq T$.

(i) $\mathrm{Param}(\mathrm{par}) \to (n := n_1(n_2+T+1), \mathbf{b}, \alpha)$, where we let $\mathbf{b} := (\{b_{i,j}\}_{i\in[1,n_1],j\in[1,n_2]}, \{b'_{i,t}\}_{i\in[1,n_1],t\in[0,T]})$ be the common variables, i.e. $b_{i,j}, b'_{i,t} \in_R \mathbb{Z}_p$ for each i, j, t, and $\alpha \in_R \mathbb{Z}_p$.

(ii) $\mathrm{EncC}(\mathbf{b}, (A,\rho)) \to (\mathbf{c}(\mathbf{s},\mathbf{b}) := (c_1, c_2)$, where $\mathbf{s} = s \in_R \mathbb{Z}_p$, $c_1 = s$ and

$$c_2 = s\left(\sum_{i\in[1,n_1],j\in[1,n_2]} a_{i,j}b_{i,j} + \sum_{i\in[1,n_1],t\in[0,T]} \rho(i)^t b'_{i,t}\right).$$

(iii) $\mathrm{EncK}(\alpha, \mathbf{b}, \mathcal{S}) \to (\mathbf{k}(\alpha,\mathbf{r},\mathbf{b}) := (\{k_{1,i}, k_{2,i,j}, k_{3,i,\ell,j}, k_{4,i,y}, k_{5,i,\ell,t}\}_{\substack{i,\ell\in[1,n_1],i\neq\ell,\\ j\in[1,n_2],t\in[0,T],\\ y\in\mathcal{S}}}),$

where $\mathbf{r} = (r_1, ..., r_{n_1}, v_2, ..., v_{n_2}) \in_R \mathbb{Z}_p^{n_1+n_2-1}$, $v_1 = \alpha$ and

$$k_{1,i} = r_i, \quad k_{2,i,j} = r_ib_{i,j} - v_j, \quad k_{3,i,\ell,j} = r_ib_{\ell,j}$$
$$k_{4,i,y} = r_i \sum_{t\in[0,T]} y^t b'_{i,t}, \quad k_{5,i,\ell,t} = r_ib'_{\ell,t}.$$

(iv) $\mathrm{Pair}((A,\rho),\mathcal{S}) \to \mathbf{E})$. Let $\mathrm{Y} = \{i \in [1,n_1] : \rho(i) \in \mathcal{S}\}$, and $\{\varepsilon_i \in \mathbb{Z}_N\}_{i\in\mathrm{Y}}$ be such that $\sum_{i\in\mathrm{Y}} \varepsilon_iA_i = (1,0,...,0)$, where A_i denotes the i-th row of A. We write matrix $\mathbf{E}$ as a set of entries $E_{t,\ell}$ where t matches the indices of $\mathbf{k}$ and ℓ matches the indices of $\mathbf{c}$. Then we set

$$E_{(1,i),2} = \varepsilon_i, \quad E_{(2,i,j),1} = -\varepsilon_ia_{i,j}, \quad E_{(3,i,\ell,j),1} = -\varepsilon_ia_{\ell,j},$$
$$E_{(4,i,\rho(i)),1} = -\varepsilon_i, \quad E_{(5,i,\ell,t),1} = -\varepsilon_i\rho(\ell)^t,$$

for all $i, \ell \in [1,n_1], \ell \neq i, j \in [1,n_2], t \in [0,T]$. The rest of the entries is 0.

Note that, technically, our definitions of $\mathbf{b}$ and $\mathbf{k}$ (and in a way, also $\mathbf{E}$) are not written as vectors but sets. However, we can easily map the sets to vectors such that they do conform to the vector notations as used in Definition 24. But for notation purposes, we rather define them as sets as it makes them more readable.

Lemma 32 *This pair encoding scheme satisfies the correctness property of Def. 25.*

Proof. We have

$$\sum_{i\in\mathrm{Y}} E_{(1,i),2}k_{1,i}c_2 + \sum_{\substack{i\in\mathrm{Y},\\ j\in[1,n_2]}} E_{(2,i,j),1}k_{2,i,j}c_1 + \sum_{\substack{i\in\mathrm{Y},\ell\in[1,n_1],\ell\neq i,\\ j\in[1,n_2]}} E_{(3,i,\ell,j),1}k_{3,i,\ell,j}c_1$$
$$+ \sum_{i\in\mathrm{Y}} E_{(4,i,\rho(i)),1}k_{4,i,\rho(i)}c_1 + \sum_{\substack{i\in\mathrm{Y},\ell\in[1,n_1],\ell\neq i,\\ t\in[0,T]}} E_{(5,i,\ell,t),1}k_{5,i,\ell,t}c_1$$
$$= \sum_{i\in\mathrm{Y}} \varepsilon_i\left(r_ic_2 - c_1\left(\sum_{\substack{\ell\in[1,n_1],\\ j\in[1,n_2]}} a_{\ell,j}r_ib_{\ell,j} - \sum_j a_{i,j}v_j + \sum_{\substack{\ell\in[1,n_1],\\ j\in[1,n_2]}} r_i\rho(\ell)^tb'_{\ell,t}\right)\right)$$
$$= \alpha s + \sum_{i\in\mathrm{Y}} \varepsilon_ir_i\left(s\left(\sum_{\ell\in[1,n_1],j\in[1,n_2]} a_{\ell,j}b_{\ell,j} + \sum_{\ell\in[1,n_1],t\in[0,T]} \rho(\ell)^tb'_{\ell,t}\right)\right.$$
$$\left.- s\left(\sum_{\ell\in[1,n_1],j\in[1,n_2]} a_{\ell,j}b_{\ell,j} + \sum_{\ell\in[1,n_1],t\in[0,T]} \rho(\ell)^tb'_{\ell,t}\right)\right) = \alpha s,$$

because $\sum_{i\in\Upsilon, j\in[1,n_2]} c_1\varepsilon_i a_{i,j} v_j = \alpha s$. □

Now we want to know whether this pair encoding scheme is secure. First, we observe that the pair encoding scheme is not perfectly secure, because the statistical distance of $\{(\mathbf{c}(\mathbf{s},\mathbf{b}), \mathbf{k}(\alpha,\mathbf{r},\mathbf{b})) : \mathbf{r},\mathbf{b},\mathbf{s} \in \mathbb{Z}_p^{n+n_1+n_2}\}$ and $\{(\mathbf{c}(\mathbf{s},\mathbf{b}), \mathbf{k}(0,\mathbf{r},\mathbf{b})) : \mathbf{r},\mathbf{b},\mathbf{s} \in \mathbb{Z}_p^{n+n_1+n_2}\}$ for $\alpha \neq 0$ is 1, which is maximal (and therefore the distributions do not overlap anywhere). However, we can prove the following:

Lemma 33 *This pair encoding scheme is relaxed perfectly secure. The* Samp *algorithm depends on* (A, ρ)*, which means that the CP-ABE scheme that follows under Construction 29 is semi-adaptively secure.*

Proof sketch. The full proof of this is in [AC16], but we will give a brief explanation of the steps that are taken in the proof. First, we define the Samp algorithm, which we do as follows:

- $\mathrm{Samp}(d,(A,\rho),\mathcal{S}) \to (\hat{\mathbf{b}}_d = \{\hat{b}_{i,j}\}_{i\in[1,n_1],j\in[1,n_2]}, \{\hat{b}'_{i,t}\}_{i\in[1,n_1],t\in[0,T]})$. On input $d \in [1, n_1+n_2-1]$, access structure (A,ρ) and set of attributes $\mathcal{S}$, the algorithm first checks whether $d \in [1,n_1]$ or not. If not, then the output is all-zero. Otherwise, it checks whether $\rho(d) \in \mathcal{S}$ or not. If $\rho(d) \notin \mathcal{S}$, then $(\hat{b}_{d,1}, ..., \hat{b}_{d,n_2}) \in_R \mathbb{Z}_p^{n_2}$. If $\rho(d) \in \mathcal{S}$, then $(\hat{b}_{d,1}, ..., \hat{b}_{d,n_2}) \in_R \mathbb{Z}_p^{n_2}$ but with the extra requirement that $\sum_{j\in[1,n_2]} a_{d,j}\hat{b}_{d,j} = 0$. Then the output of Samp is $\{\hat{b}_{i,j}\}_{i,j}, \{\hat{b}'_{i,t}\}_{i,t}$ such that $\hat{b}_{i,j} = 0$ for all $i \neq d$ and $j \in [1,n_2]$, and $\hat{b}'_{i,t} = 0$ for all i, t.

It is obvious that Samp depends on the access structure (A,ρ), which means that the encryption scheme that follows under Construction 29 is proven semi-adaptively secure.

For $d \in [1,n_1]$, we have

$$(\mathbf{c}(\mathbf{s},\mathbf{b}), \mathbf{k}(0,\mathbf{r}_d,\mathbf{b}+\hat{\mathbf{b}}_d)) = \left(s, s\left(\sum_{i,j} a_{i,j}b_{i,j} + \sum_{i,t}\rho(i)^t b'_{i,t}\right), \mathbf{k}(0,\mathbf{r}_d,\mathbf{b}+\hat{\mathbf{b}}_d)\right),$$

for which $\mathbf{k}(0,\mathbf{r}_d,\mathbf{b}+\hat{\mathbf{b}}_d))$ consists of mostly 0-entries, except for $k_{1,d}, k_{2,d,j}, k_{3,d,\ell,j}, k_{4,d,y}$ and $k_{5,d,\ell,t}$ for all $\ell \in [1,n_1], \ell \neq d, j \in [1,n_2]$ and $t \in [0,T]$. Note that only $k_{2,d,j} = r_d(b_{d,j} + \hat{b}_{d,j}) - v_j$, has some extra addition of noise, the rest of the entries is unaffected by this noise. The idea is that this extra randomness 'cancels out' the α in $\mathbf{k}(\alpha,\mathbf{r}_d,\mathbf{b}+\hat{\mathbf{b}}_d)$, because $\alpha + \hat{b}_{d,1}$ is perfectly indistinguishable from $\hat{b}_{d,1}$.

However, we still need to show that, for each such d, adding this noise cannot be detected by the attacker. First, we observe that, because $b_{i,j}$ and $\hat{b}_{i,j}$ are both uniformly distributed, no attacker can distinguish $\mathbf{k}(0,\mathbf{r}_d,\mathbf{b}))$ from $\mathbf{k}(0,\mathbf{r}_d,\mathbf{b}+\hat{\mathbf{b}}_d))$ by just observing the secret key part. But the attacker can also observe the ciphertext encoding, and in particular the second part, which is

$$s\left(\sum_{i,j} a_{i,j}b_{i,j} + \sum_{i,t}\rho(i)^t b'_{i,t}\right).$$

Let us divide this equation into two parts: $\Psi \subseteq [1, n_1]$ such that $\Psi = \{i \in [1, n_1] : \rho(i) \in \mathcal{S}\}$ and its complement $\overline{\Psi} = [1, n_1] \setminus \Psi$. Then we have

$$\begin{aligned} & s\left(\sum_{i\in\Psi}\left(\sum_j a_{i,j}b_{i,j} + \sum_t \rho(i)^t b'_{i,t}\right) + \sum_{i\in\overline{\Psi}}\left(\sum_j a_{i,j}b_{i,j} + \sum_t \rho(i)^t b'_{i,t}\right)\right) \\ & = s\left(\sum_{i\in\Psi}\left(\sum_j a_{i,j}(\bar{b}_{i,j} - \hat{b}_{i,j}) + k_{4,i,\rho(i)}\right) + \sum_{i\in\overline{\Psi}} \text{randomremainder}_i\right) \\ & = s\left(\sum_{i\in\Psi}\left(\sum_j a_{i,j}\bar{b}_{i,j} + k_{4,i,\rho(i)}\right) + \sum_{i\in\overline{\Psi}} \text{randomremainder}_i\right) \\ & = s\left(\sum_{i\in\Psi}\left(\sum_j a_{i,j}\frac{k_{2,i,j}}{k_{1,i}} + k_{4,i,\rho(i)}\right) + \sum_{i\in\overline{\Psi}} \text{randomremainder}_i\right) \end{aligned}$$

where $\bar{b}_{i,j} = b_{i,j} + \hat{b}_{i,j}$ for all i and j (but $\hat{b}_{i,j} = 0$ for $i \neq d$). Note that the equation indeed holds because for $i \neq d$, all of the steps in the equation trivially hold, and for d we observe: if $d \in \Psi$, then $\sum_j a_{d,j}\hat{b}_{d,j} = 0$, and if $d \notin \Psi$, then it is part of the random remainder (which behaves randomly because we do not have $k_{4,d,\rho(d)}$ and therefore $b'_{d,t}$ behave completely randomly, because we assumed $|\mathcal{S}| \leq T$, and $k_{4,d,\rho(d)}$ denotes a T-degree polynomial). Hence, adding noise $\hat{\mathbf{b}}_d$ is not observable by an attacker. Note that we have to require that $\sum_j a_{d,j}\hat{b}_{d,j} = 0$, because the noisy keys have to be able to decrypt normal ciphertexts for valid sets of attributes.

The second part of the proof shows that

$$\left\{\left(\mathbf{c}(\mathbf{s},\mathbf{b}), \sum_{d\in[1,m_2]} \mathbf{k}(0, \mathbf{r}_d, \mathbf{b} + \hat{\mathbf{b}}_d)\right)\right\} \cong \left\{\left(\mathbf{c}(\mathbf{s},\mathbf{b}), \sum_{d\in[1,m_2]} \mathbf{k}(\alpha, \mathbf{r}_d, \mathbf{b} + \hat{\mathbf{b}}_d)\right)\right\}$$

holds, by arguing that the extra noise $\hat{\mathbf{b}}_d$ 'cancels out' the α. □

Remark: We observe that, indeed, the security proof of PES is much shorter than the security proof of CP-ABE: the proof as given in [AC16] is roughly two pages in LNCS style, whereas the security proof of the generic CP-ABE was eight pages long.

In [AC17a], Agrawal and Chase prove that the encryption scheme that follows from this PES under Construction 29 is fully secure. To prove this, Agrawal and Chase introduce the notion of *symbolic security* in pair encoding schemes. Whereas [AC16] proves the CP-ABE scheme semi-adaptively secure under the SXDH assumption, [AC17a] proves the same CP-ABE scheme to be fully secure under a q-type assumption called the q-ratio$_{\text{DSG}}$ assumption.

B.3 Constructing the scheme

Now we can apply the generic construction in Construction 29 to the pair encoding scheme defined in Section B.2.1.

We will do this by considering each algorithm of the generic construction, and observe which algorithms and variables of the dual system group and pair encoding scheme we need in order to realize this algorithm. Naturally, we will start with the Setup algorithm, and work our way through the other algorithms until we finish with the Decrypt algorithm. At last, we will give the complete construction, without using invocations of dual system group or pair encoding algorithms.

Recall that in Section B.2.1, we had assumed that the ciphertext encodings take LSSS access structures as inputs, where the matrices are restricted in the sense that the size is fixed, i.e. we only consider $n_1 \times n_2$ matrices. The key encodings take sets of attributes $\mathcal{S}$ as input, for which holds that the size may not be larger than T.

B.3.1 The Setup algorithm

Running Param(par) yields $n = (n_1(n_2 + T + 1), \mathbf{b}, \alpha)$, where $\alpha \in_R \mathbb{Z}_p$ and $\mathbf{b} = (\{b_{i,j}\}_{i\in[1,n_1],j\in[1,n_2]}, \{b'_{i,t}\}_{i\in[1,n_1],t\in[0,T]}$. Note that instead of using α, we use two randoms $\alpha_1, \alpha_2 \in_R \mathbb{Z}_p$.

Then, we run $\mathrm{SampP}(1^\lambda, 1^n)$, which yields tuple $(p, \mathbb{G}, \mathbb{H}, \mathbb{G}_T, e)$, where p is a prime of at least λ bits, and $\mathbb{G} = G \times G$ and $\mathbb{H} = H \times H$, where G and H denote two cyclic groups of prime order, with generators g and h. Then, e is a non-degenerate bilinear mapping $e : G \times H \to G_T$, and $\mathbb{G}_T = G_T$. We also obtain $\mu : \mathbb{H} \to \mathbb{G}_T$, defined as $\mu(h^{\mathbf{k}}) = e(g^{\mathbf{D}}, h^{\mathbf{k}})$, where $\mathbf{k} \in \mathbb{Z}_p^2$.

Moreover, we obtain $\mathbf{g}_0, \{\mathbf{g}_{i,j}\}_{i\in[1,n_1],j\in[1,n_2]}, \{\mathbf{g}'_{i,t}\}_{i\in[1,n_1],t\in[0,T]}$ such that $\mathbf{g}_0 = g^{\mathbf{D}}$, $\mathbf{g}_{i,j} = g^{\mathbf{D}_{i,j}}$ and $\mathbf{g}'_{i,t} = g^{\mathbf{D}'_{i,t}}$ for all $i \in [1, n_1], j \in [1, n_2]$ and $t \in [0, T]$. Here, $\mathbf{D}, \mathbf{D}_{i,j}, \mathbf{D}'_{i,t}$ are defined as in Section B.1.1. We also obtain $\mathbf{h}_0, \{\mathbf{h}_{i,j}\}_{i\in[1,n_1],j\in[1,n_2]}$, $\{\mathbf{h}'_{i,t}\}_{i\in[1,n_1],t\in[0,T]}$, which we define similarly. We compute the master secret key as $\mathbf{MSK} = (h^{\alpha_1}, h^{\alpha_2})$. Our master public key is

$$\mathrm{MPK} = \Big((p, \mathbb{G}, \mathbb{H}, \mathbb{G}_T, e); \mu; \mu(\mathbf{MSK}); \mathbf{g}_0, \{\mathbf{g}_{i,j}\}_{i,j}, \{\mathbf{g}'_{i,t}\}_{i,t}; \mathbf{h}_0, \{\mathbf{h}_{i,j}\}_{i,j}, \{\mathbf{h}'_{i,t}\}_{i,t}\Big).$$

B.3.2 The Encrypt algorithm

The inputs are master public key MPK, access structure $\mathbb{A} = (A, \rho)$ and message $M \in \mathbb{G}_T$. We first run $\mathrm{EncC}(\mathbf{b}, \mathbb{A})$, such that we obtain ciphertext encoding $\mathbf{c}(\mathbf{s}, \mathbf{b}) = (c_1, c_2)$. In this case, $\mathbf{s} = (s) \in_R \mathbb{Z}_p$, $c_1 = s$ and $c_2 = s(\sum_{i\in[1,n_1],j\in[1,n_2]} a_{i,j} b_{i,j} + \sum_{i\in[1,n_1],t\in[0,T]} \rho(i)^t y^t)$, where $a_{i,j}$ denote the entries of A, i.e. $A_{ij} = a_{i,j}$ for each $i \in [1, n_1], j \in [1, n_2]$. Because $\mathbf{s}$ only consists of one entry, we only need to draw one sample from $\mathbb{G}$ with SampG, for which we let s be the random that is used in the sampling, i.e.

$$(\mathbf{g}_0^s, \{\mathbf{g}_{i,j}^s\}_{i,j}, \{(\mathbf{g}'_{i,t})^s\}_{i,t}) \leftarrow \mathrm{SampG}(\mathrm{PP}; s).$$

Then we define $\mathbf{CT}_1 = \mathbf{g}_0^s$ and $\mathbf{CT}_2$ as

$$\mathbf{CT}_2 = \prod_{i\in[1,n_1],j\in[1,n_2]} \mathbf{g}_{i,j}^{s a_{i,j}} \prod_{i\in[1,n_1],t\in[0,T]} (\mathbf{g}'_{i,t})^{s\rho(i)^t}.$$

Note that both $\mathbf{CT}_1$ and $\mathbf{CT}_2$ denote elements in $\mathbb{G} = G \times G$, and therefore both consist of two elements in G.

The last part of the ciphertext is $\mathrm{CT}_3 = M \cdot \mathrm{SampGT}(\mu(\mathbf{MSK}); s) = M \cdot e(g^{\mathbf{D}}, \mathrm{PK})^s = M \cdot e(\mathbf{g}_0, \mathrm{PK})^s$. So our output is the ciphertext

$$\mathrm{CT} = ((A, \rho), \mathbf{CT}_1, \mathbf{CT}_2, \mathrm{CT}_3),$$

which consists of an access structure and five group elements of order p.

B.3.3 The KeyGen algorithm

The key generation takes the master secret key $\mathbf{MSK}$ and set of attributes $\mathcal{S}$ as inputs, for which we assume that $|\mathcal{S}| \leq T$. We run $\mathrm{EncK}(\alpha_1, \mathbf{b}, \mathcal{S})$ to obtain key encoding $\mathbf{k}(\alpha_1, \mathbf{r}, \mathbf{b})$, where $\mathbf{r} = (r_1, ..., r_{n_1}, v_2, ..., v_{n_2}) \in_R \mathbb{Z}_p^{n_1+n_2-1}$ and $v_1 = \alpha_1$, such that $\mathbf{k}(\alpha_1, \mathbf{r}, \mathbf{b})$ has entries

$$k_{1,i} = r_i, \quad k_{2,i,j} = r_i b_{i,j} - v_j, \quad k_{3,i,\ell,j} = r_i b_{\ell,j}$$
$$k_{4,i,y} = r_i \sum_{t \in [0,T]} y^t b'_{i,t}, \quad k_{5,i,\ell,t} = r_i b'_{\ell,t}.$$

Then for each r_i and v_j, we sample from $\mathbb{H}$ with SampH, for which we will be using the random elements $\mathbf{r} = (r_1, ..., r_{n_1}, v_2, ..., v_{n_2})$ for better readability, i.e.

$$(\mathbf{h}_0^{r_1}, \{\mathbf{h}_{i,j}^{r_1}\}_{i,j}, \{(\mathbf{h}'_{i,t})^{r_1}\}_{i,t}) \leftarrow \mathrm{SampH}(\mathrm{PP}; r_1), ...,$$
$$(\mathbf{h}_0^{v_{n_2}}, \{\mathbf{h}_{i,j}^{v_{n_2}}\}_{i,j}, \{(\mathbf{h}'_{i,t})^{v_{n_2}}\}_{i,t}) \leftarrow \mathrm{SampH}(\mathrm{PP}; v_{n_2}).$$

The secret key is denoted as $\mathrm{SK} = (\mathcal{S}, \{\mathbf{SK}_{1,i}, \mathbf{SK}_{2,i,j}, \mathbf{SK}_{3,i,\ell,j}, \mathbf{SK}_{4,i,y}, \mathbf{SK}_{5,i,\ell,t}\}_{i,\ell,j,y,t})$, such that

$$\mathbf{SK}_{1,i} = \mathbf{h}_0^{r_i} \qquad \mathbf{SK}_{2,i,j} = \begin{cases} \mathbf{h}_{i,j}^{r_i} / \mathbf{h}_0^{v_j} & \text{if } j > 1 \\ \mathbf{h}_{i,1}^{r_i} / \mathbf{MSK} & \text{if } j = 1 \end{cases}$$
$$\mathbf{SK}_{3,i,\ell,j} = \mathbf{h}_{\ell,j}^{r_i} \qquad \mathbf{SK}_{4,i,y} = \textstyle\prod_{t \in [0,T]} (\mathbf{h}'_{i,t})^{r_i y^t}$$
$$\mathbf{SK}_{5,i,\ell,t} = (\mathbf{h}'_{\ell,t})^{r_i}.$$

B.3.4 The Decrypt algorithm

The decryption algorithm takes master public parameters MPK, the secret key $\mathrm{SK} = (\mathcal{S}, \{\mathbf{SK}_{1,i}, \mathbf{SK}_{2,i,j}, \mathbf{SK}_{3,i,\ell,j}, \mathbf{SK}_{4,i,y}, \mathbf{SK}_{5,i,\ell,t}\}_{i,\ell,t,y})$ and the ciphertext $\mathrm{CT} = ((A, \rho), \mathbf{CT}_1, \mathbf{CT}_2, \mathrm{CT}_3)$ as inputs, such that $\mathcal{S}$ denotes the set of attributes and (A, ρ) is the access structure that was used to encrypt the ciphertext. We run $\mathrm{Pair}((A, \rho), \mathcal{S})$, such that we obtain for $\Upsilon \subseteq [1, n_1]$ the set of rows corresponding to the set of attributes $\mathcal{S}$, and $\{\varepsilon_i\}_{i \in \Upsilon}$ such that $\sum_{i \in \Upsilon} \varepsilon_i A_i = (1, 0, ..., 0)$. Then Pair yields $\mathbf{E}$ such that

$$E_{(1,i),2} = \varepsilon_i, \quad E_{(2,i,j),1} = -\varepsilon_i a_{i,j}, \quad E_{(3,i,\ell,j),1} = -\varepsilon_i a_{\ell,j},$$
$$E_{(4,i,\rho(i)),1} = -\varepsilon_i, \quad E_{(5,i,\ell,t),1} = -\varepsilon_i \rho(\ell)^t,$$

for all $i \in \Upsilon, \ell \in [1, n_1], \ell \neq i, j \in [1, n_2], t \in [0, T], y \in \mathcal{S}$.

The decryption then works as follows:

$$\mathrm{CT}_3 \cdot \left(\prod_{x \in \mathcal{I}} e(\mathbf{CT}_1, \mathbf{SK}_x^{\mathbf{E}_{x,1}}) \cdot \prod_{x \in \mathcal{I}} e(\mathbf{CT}_2, \mathbf{SK}_x^{\mathbf{E}_{x,2}}) \right)^{-1},$$

where $\mathcal{I} = \{(1, i), (2, i, j), (3, i, \ell, j), (4, i, y), (5, i, \ell, t)\}_{i,\ell,t,y}$ denotes the set of indices for which $\mathbf{E}_{x,1}$ or $\mathbf{E}_{x,2}$ is not equal to 0.

More specifically, we compute $C_2 = e(\mathbf{CT}_2, \prod_{x \in \mathcal{I}} \mathbf{SK}_x^{\mathbf{E}_{x,2}}) = e(\mathbf{CT}_2, \prod_{i \in \Upsilon} \mathbf{SK}_{1,i}^{\varepsilon_i})$, and $C_1 = e(\mathbf{CT}_1, \prod_{x \in \mathcal{I}} \mathbf{SK}_x^{\mathbf{E}_{x,1}})$ by first computing for all $i \in \Upsilon$:

$$\mathbf{S}_{2,i} = \prod_{j \in [1, n_2]} \mathbf{SK}_{2,i,j}^{a_{i,j}}, \qquad \mathbf{S}_{3,i} = \prod_{\ell \in [1, n_1], \ell \neq i, j \in [1, n_2]} \mathbf{SK}_{3,i,\ell,j}^{a_{\ell,j}},$$
$$\mathbf{S}_{4,i} = \mathbf{SK}_{4,i,\rho(i)}, \qquad \mathbf{S}_{5,i} = \prod_{\ell \in [1, n_1], \ell \neq i, t \in [0,T]} \mathbf{SK}_{5,i,\ell,t}^{\rho(\ell)^t}$$

and then $C_1 = e\left(\mathbf{CT}_1, \prod_{i \in \Upsilon} (\mathbf{S}_{2,i} \mathbf{S}_{3,i} \mathbf{S}_{4,i} \mathbf{S}_{5,i})^{\varepsilon_i}\right)$. Finally, we retrieve M by computing

$$\mathrm{CT}_3 \cdot C_1 / C_2.$$

B.3.5 The complete construction

We can combine the previously explained algorithms into a more compact construction that resembles the descriptions of other ABE schemes, such as the ones that we have shown in Definitions 19 and 20.

Definition 34 (The [AC16; CW14a] scheme) *We represent access structures by LSSS matrices, denoted as $\mathbb{A} = (A, \rho)$, where A is a $n_1 \times n_2$ matrix with entries in $\mathbb{Z}_p$, where p is a prime of at least λ bits, and $\rho : [1, n_1] \to \mathcal{U}$, which maps the rows of A to attributes, where $\mathcal{U} = \mathbb{Z}_p$ denotes the (large) universe of attributes. We denote A_i as the i-th row of A and $a_{i,j} = A_{ij}$ as the j-th element of the i-th row. For set of attributes $\mathcal{S} \subseteq \mathcal{U}$ we define $\Upsilon = \{i : i \in [1, n_1], \rho(i) \in \mathcal{S}\}$ to be the set of indices of rows in A associated with $\mathcal{S}$. Furthermore, for this scheme, we define $T \in \mathbb{N}$ to be the maximum number of attributes a user is allowed to possess, i.e. $|\mathcal{S}| \leq T$.*

(i) $\mathrm{Setup}(1^\lambda)$: *The* Setup *algorithm takes the security parameter λ as input, and generates (p, G, H, G_T, g, h), where p is a prime of at least λ bits, and G and H are cyclic groups of order p, with generators $g \in G$ and $h \in H$. We also define non-degenerate bilinear mapping $e : G \times H \to G_T$. Let $\mathbf{B} \in_R \mathbb{Z}_p^{2\times 2}$ be an invertible matrix, and set $\mathbf{B}^* = (\mathbf{B}^{-1})^t$ be the transposed inverse of $\mathbf{B}$. Furthermore, pick $\mathbf{A}_{i,j}, \mathbf{A}'_{i,t} \in_R \mathbb{Z}_p^{2\times 2}$ for each $i \in [1, n_1], j \in [1, n_2], t \in [0, T]$, and let $\mathbf{R} \in \mathbb{Z}_p^{2\times 2}$ be an invertible diagonal matrix with $\mathbf{R}_{22} = 1$. Define*

$$\begin{aligned}
&\mathbf{D} := \pi_L(\mathbf{B}), \quad &&\mathbf{D}_{i,j} := \pi_L(\mathbf{B}\mathbf{A}_{i,j}), \quad &&\mathbf{D}'_{i,t} := \pi_L(\mathbf{B}\mathbf{A}'_{i,t})\\
&\mathbf{D}^* := \pi_L(\mathbf{B}^*\mathbf{R}), &&\mathbf{D}^*_{i,j} := \pi_L(\mathbf{B}^*\mathbf{A}^t_{i,j}\mathbf{R}), &&\mathbf{D}'^*_{i,t} := \pi_L(\mathbf{B}^*(\mathbf{A}'_{i,t})^t\mathbf{R})
\end{aligned}$$

which we use to define

$$\begin{aligned}
\mathbf{g}_0 = g^{\mathbf{D}}, \{\mathbf{g}_{i,j} = g^{\mathbf{D}_{i,j}}, \mathbf{g}'_{i,t} = g^{\mathbf{D}'_{i,t}}\}_{i,j,t}\\
\mathbf{h}_0 = h^{\mathbf{D}^*}, \{\mathbf{h}_{i,j} = h^{\mathbf{D}^*_{i,j}}, \mathbf{h}'_{i,t} = h^{\mathbf{D}'^*_{i,t}}\}_{i,j,t}
\end{aligned}.$$

We also generate master secret key $\mathbf{MSK} = (h^{\alpha_1}, h^{\alpha_2})$ *for $\alpha_1, \alpha_2 \in_R \mathbb{Z}_p$. Then the public parameters are*

$$\mathrm{MPK} = \left(p, G, H, G_T, e, e(\mathbf{g}_0, \mathbf{MSK}), \mathbf{g}_0, \{\mathbf{g}_{i,j}\}_{i,j}, \{\mathbf{g}'_{i,t}\}_{i,t}; \mathbf{h}_0, \{\mathbf{h}_{i,j}\}_{i,j}, \{\mathbf{h}'_{i,t}\}_{i,t}\right).$$

(ii) $\mathrm{Encrypt}(\mathrm{MPK}, (A, \rho), M)$: *The* Encrypt *algorithm takes the master public key as input, as well as the access structure (A, ρ) and message $M \in G_T$. Let $s \in_R \mathbb{Z}_p$. The output ciphertext is* $\mathrm{CT} = ((A, \rho), \mathbf{CT}_1, \mathbf{CT}_2, \mathrm{CT}_3)$, *such that*

$$\begin{aligned}
\mathbf{CT}_1 &= \mathbf{g}_0^s\\
\mathbf{CT}_2 &= \prod_{i\in[1,n_1], j\in[1,n_2]} \mathbf{g}_{i,j}^{s a_{i,j}} \prod_{i\in[1,n_1], t\in[0,T]} (\mathbf{g}'_{i,t})^{s\rho(i)^t}\\
\mathrm{CT}_3 &= M \cdot e(\mathbf{g}_0, \mathbf{MSK})^s.
\end{aligned}$$

(iii) $\mathrm{KeyGen}(\mathbf{MSK}, \mathcal{S})$: *The* KeyGen *algorithm takes the master secret key and the set of attributes $\mathcal{S}$ that the user possesses, for which we require $|\mathcal{S}| \leq T$, as inputs, and generates $r_i, v_j \in_R \mathbb{Z}_p$ for all $i \in [1, n_1]$ and $j \in [2, n_2]$. The secret key is defined as*

$$\mathrm{SK} = (\mathcal{S}, \{\mathbf{SK}_{1,i}, \mathbf{SK}_{2,i,j}, \mathbf{SK}_{3,i,\ell,j}, \mathbf{SK}_{4,i,y}, \mathbf{SK}_{5,i,\ell,t}\}_{\substack{i,\ell\in[1,n_1], i\neq\ell,\\ j\in[1,n_2], y\in\mathcal{S}, t\in[0,T]}}),$$

for which we have

$$\begin{aligned}
\mathbf{SK}_{1,i} &= \mathbf{h}_0^{r_i} \\
\mathbf{SK}_{2,i,j} &= \begin{cases} \mathbf{h}_{i,j}^{r_i}/\mathbf{h}_0^{v_j} & \text{if } j > 1 \\ \mathbf{h}_{i,1}^{r_i}/\mathbf{MSK} & \text{if } j = 1 \end{cases} \\
\mathbf{SK}_{3,i,\ell,j} &= \mathbf{h}_{\ell,j}^{r_i} \\
\mathbf{SK}_{4,i,y} &= \prod_{t\in[0,T]} (\mathbf{h}'_{i,t})^{r_i y^t} \\
\mathbf{SK}_{5,i,\ell,t} &= (\mathbf{h}'_{\ell,t})^{r_i}.
\end{aligned}$$

(iv) Decrypt(MPK, SK, CT) : *The* Decrypt *algorithm takes the master public key* MPK*, the secret key* SK *related to* $\mathcal{S}$ *and ciphertext* CT *encrypted under* (A,ρ) *as input, and then determines for the set of attributes* $\Upsilon \subseteq \mathcal{S}$ *that correspond to the rows of* A *the set* $\{\varepsilon_i\}_{i\in\Upsilon}$ *such that* $\sum_{i\in\Upsilon}\varepsilon_i A_i = (1,0,...,0)$*. Then it computes* $C_2 = e(\mathbf{CT}_2, \prod_{i\in\Upsilon}\mathbf{SK}_{1,i}^{\varepsilon_i})$ *and for all* $i \in \Upsilon$*, it computes*

$$\mathbf{S}_{2,i} = \prod_{j\in[1,n_2]} \mathbf{SK}_{2,i,j}^{a_{i,j}}, \qquad \mathbf{S}_{3,i} = \prod_{\ell\in[1,n_1],\ell\neq i,j\in[1,n_2]} \mathbf{SK}_{3,i,\ell,j}^{a_{\ell,j}},$$

$$\mathbf{S}_{4,i} = \mathbf{SK}_{4,i,\rho(i)}, \qquad \mathbf{S}_{5,i} = \prod_{\ell\in[1,n_1],\ell\neq i,t\in[0,T]} \mathbf{SK}_{5,i,\ell,t}^{\rho(\ell)^t},$$

then sets $C_1 = e\left(\mathbf{CT}_1, \prod_{i\in\Upsilon}(\mathbf{S}_{2,i}\mathbf{S}_{3,i}\mathbf{S}_{4,i}\mathbf{S}_{5,i})^{\varepsilon_i}\right)$*, and retrieves the plaintext by computing* $\mathrm{CT}_3 \cdot C_1/C_2$*.*

Correctness: The correctness of this scheme follows directly from the correctness of the generic construction in 29, though the full proof of correctness can be found in Appendix D.

B.3.6 Storage and computational costs

Now that we have given a clear definition of the [AC16; CW14a] scheme, we can analyze the storage and computational costs.

For the storage costs, we have that the secret key associated with $\mathcal{S}$ consists of $2n_1(n + |\mathcal{S}| - T) \leq 2n_1 n$ group elements and will therefore be at most $2n_1 n\lambda$ bits long. The ciphertext associated with any access structure always consists of five group elements (four in G and one in G_T).

In Table B.1, the computational costs are depicted. In this analysis, we have taken LSSS matrices A that are generated with the algorithm that was described by Lewko and Waters in [LW10]. Because all of the entries in A are in $\{1,-1,0\}$ as well as ε_i for all i, most of the computations that seem to denote exponentations are actually multiplications or divisions (e.g. the computation of $\mathbf{S}_{2,i}$).

As we can see in Table B.1, the total number of pairing operations is very limited, which is an advantage in comparison with other schemes. However, the number of exponentiations in key generation is still quite large, especially compared to other schemes, which makes it interesting to consider whether we can move part of the key generation away from the key generation authority to either semi-trusted proxies or the users themselves (a large part of the secret key is not related to the attributes that the user possesses). Moreover, in some of the algorithms, we might be able to apply optimization steps. For instance, in the decryption we will be able to reduce the number of exponentatiations, as $\prod_i \mathbf{S}_{5,i}^{\varepsilon_i}$ can be rewritten as $\prod_{\ell\neq i,j}(\prod_{i\in\Upsilon}\mathbf{SK}_{5,i,\ell,j})^{\rho(\ell)^t}$,

Alg.	**Pair.**	**Exp.**	**Mult. & div.**
Setup	2	$2n+2$ in G, $2n+4$ in H	✗
Encrypt	✗	$2n_1(T+1)+4$ in G, 1 in G_T	$2n-2$ in G, 1 in G_T
KeyGen	✗	$2n_1(n+T(\|\mathcal{S}\|-1)+\mathcal{S})$ $+2n_2-2$ in H	$2(n_1n_2+T\|\mathcal{S}\|)$ in H
Decrypt	4	$2\|\Upsilon\|(n_1-1)(T+1)$ in H	$2\|\Upsilon\|(n_1n_2+4)$ in H, 2 in G_T

TABLE B.1: Computational costs

which reduces the number of exponentiations by a factor $1/|\Upsilon|$. However, this will not be the focus of this work, as we are already fairly satisfied with the results and the possibilities of optimizing the algorithms.

B.4 Discussion

As we have shown, the [AC16; CW14a] scheme is semi-adaptively secure under the SXDH assumption (and fully secure under the q-ratio$_{\mathrm{DSG}}$ assumption [AC17a]), which makes it the most secure CP-ABE scheme with constant-size ciphertexts.

Moreover, the scheme allows for very expressive access structures, over a large universe of attributes. Another advantage of this scheme over other schemes such as [LW11] is that this scheme allows for multi-use of attributes in the access policies, without having to compute more public and secret keys.

However, there are also some obvious disadvantages. Namely, our scheme is restricted in the size of the access matrix: the maximum number of rows is fixed in n_1, and the maximum number of columns is fixed in n_2. Another restriction is T, the maximum number of attributes that a user in the system can possess. One way to limit the problems that this might cause is to make the scheme dynamic. If a user wants to request a set of keys corresponding to the set of attributes $\mathcal{S}$, such that $|\mathcal{S}| > T$, then technically, the key generation authority could update the keys such that the new T is large enough such that $|\mathcal{S}| \leq T$ holds. The same could happen for the access structure: if some user needs a larger access structure, it can simply request the key generation authority for keys that can realize this, i.e. the key generation authority generates more $\mathbf{g}_{i,j}, \mathbf{g}'_{i,t}, \mathbf{h}_{i,j}$ and $\mathbf{h}'_{i,t}$ and sends new key sets to users that need them.

A disadvantage that cannot be mitigated this easily is the size of the secret keys associated with a set $\mathcal{S}$. Whereas the size is only linear in the number of attributes, it depends heavily on the maximum size of the access structures and the maximum size of the set of attributes as well.

But all things considered, the advantages might actually outweigh the disadvantages, and we might even be able to mitigate most of the disadvantages by applying some changes to the scheme.

As we had already briefly mentioned, we are interested in formulating a decentralized ciphertext-policy based scheme with constant-size ciphertexts, as these are – to the best of our knowledge – still nonexistent in literature, while they would provide some useful functionality in some settings such as DECODE, and other use cases in which we require the minimization of ciphertexts and the decentralization of the key generation. To this end, we will work on the decentralization of this CP-ABE scheme in Appendix C.

Appendix C

A Decentralized CP-ABE Scheme with Constant-Size Ciphertexts

As we have seen in Section 3.5, the key generation of CP-ABE schemes can be distributed across multiple authorities such that not just one central authority is responsible for the distribution of keys. In a fully decentralized scheme, we do not need any communication between the authorities after the setup. Our goal is to achieve this for the [AC16; CW14a] CP-ABE scheme that we defined in Definition 34.

For this, we need to split the key generation algorithm into parts such that an authority can only generate secret keys associated with some attribute $y \in \mathcal{S}$ if the authority manages that attribute. This might result in a slight change in the notation of our scheme, which will ripple through in the other algorithms as well, so in addition to making changes to the key generation algorithm, we might have to make changes to the other algorithms too. Especially the setup, which will be split in a global and authority setup, but possibly also the encryption and decryption algorithms will be affected by this.

In order to do this, we will first focus on the key generation algorithm, and which parts are associated with attributes, and which other parts are required to remain secret towards the authorities.

C.1 Key generation in the centralized setting

In the central authority setup, the key generation algorithm for some set of attributes $\mathcal{S}$ generates the following secret keys:

$$\begin{aligned}
\mathbf{SK}_{1,i} &= \mathbf{h}_0^{r_i} \\
\mathbf{SK}_{2,i,j} &= \begin{cases} \mathbf{h}_{i,j}^{r_i} / \mathbf{h}_0^{v_j} & \text{if } j > 1 \\ \mathbf{h}_{i,1}^{r_i} / \boxed{\mathbf{MSK}} & \text{if } j = 1 \end{cases} \\
\mathbf{SK}_{3,i,\ell,j} &= \mathbf{h}_{\ell,j}^{r_i} \\
\boxed{\mathbf{SK}_{4,i,y}} &= \prod_{t \in [0,T]} (\mathbf{h}'_{i,t})^{r_i y^t} \\
\mathbf{SK}_{5,i,\ell,t} &= (\mathbf{h}'_{\ell,t})^{r_i}.
\end{aligned}$$

We observe two things: first, we observe the occurrence of the master secret key **MSK** in the second subset of secret keys, i.e. $\mathbf{SK}_{2,i,1}$ for $i \in [1, n_1]$. This master secret key is supposed to remain secret towards users in the centralized setting. However, because knowledge of the master secret key results in being able to decrypt each ciphertext in the system, since $\mathrm{CT}_3/e(\mathbf{CT}_1, \mathbf{MSK}) = \mathrm{CT}_3/e(g_0, \mathbf{MSK})^s = M \cdot e(g_0, \mathbf{MSK})^s/e(g_0, \mathbf{MSK})^s = M$, we cannot allow that any of the authorities has knowledge of this master secret key either. So the solution to this problem would be to distribute the generation of **MSK** among all the authorities such that they can only retrieve MSK if they all collaborate (and this breaks the security of the system anyway).

Our second observation is that only one subset of secret keys depends on attribute $y \in \mathcal{S}$, namely $\{\mathbf{SK}_{4,i,y}\}_{i\in[1,n_1]}$. So our first attempt would be to put each authority $\mathcal{A}_k$ that manages attribute y in charge of distributing $\mathbf{SK}_{4,i,y}$ for all $i \in [1, n_1]$. However, we also observe that $\mathbf{SK}_{4,i,y}$ uses variables that are used for each y, so we cannot simply let authority $\mathcal{A}_k$ generate $\mathbf{h}'_{i,t}$ for all i and t. On a second note, we recall the corresponding part of the key encoding as defined in Section B.2.1, which is a polynomial: $k_{4,i,y} = r_i \sum_{t\in[0,T]} y^t b'_{i,t}$. In the key encoding, this was chosen as a polynomial such that users with attribute set $\mathcal{S}$ of size $\leq T$ could not reconstruct $r_i b'_{i,t}$, and from that also create $k_{4,i,y'}$ for y' that are not in $\mathcal{S}$. The reason for this is that they do not have enough 'points' $(y, k_{4,i,y})$ on the polynomial to reconstruct the coefficients of the polynomial with Lagrange interpolation, and therefore these 'coefficients' $r_i b'_{i,t}$ behave as random integers.

Because of the form of $k_{4,i,y}$, we can also approach the computation of those secret keys as invocations of Shamir's secret sharing scheme, of which there are several verifiable versions such as Feldman's VSS scheme, which we discussed in Section 2.4. More importantly, we have defined a decentralized VSS scheme in Definition 8 that allows for a distributed key generation algorithm in the fashion that we want to apply, because $k_{4,i,y}$ basically symbolizes what happens in the exponents, so what happens in the exponents resembles the use of polynomials. We say 'resembles' here, because in our setup we use $\prod_{t\in[0,T]}(\mathbf{h}'_{i,t})^{r_i y^t}$, which is not entirely of the form $\mathbf{h}^{P(y)}$, where $\mathbf{h}$ is some base and $P(y)$ is some polynomial. We will see, though, that we are able to write the exponent as a sum of two polynomials, in the form $(h^{a_1P_1+a_2P_2}, h^{b_1P_1+b_2P_2})$, where P_1 and P_2 are polynomials, and a_1, a_2, b_1 and b_2 are scalars in $\mathbb{Z}_p$. But in any case, this still enables us to use the decentralized VSS sheme in Definition 8.

C.2 Approach to decentralization

In the previous section, we observed that there are two subsets of secret keys that need actual decentralization in the scheme: $\mathbf{SK}_{2,i,1}$ and $\mathbf{SK}_{4,i,y}$ for all $i \in [1, n_1]$, and attributes y. We will now consider approaches to generating $\mathbf{SK}_{2,i,1}$ and $\mathbf{SK}_{4,i,y}$ in a secure, distributed fashion.

C.2.1 Generating $\mathbf{SK}_{2,i,1}$

From $\mathbf{SK}_{2,i,1}$, we had already required that the distributed generation should be done such that **MSK** remains secret, as long as at least one of the authorities is honest. To do this, we can use that $\mathbf{SK}_{2,i,1}$ is of the form $\mathbf{h}_{i,1}^{r_i}/\mathbf{MSK}$. So if each authority $\mathcal{A}_k$ generates its own share of the master secret key $\mathbf{h}_{i,1}^{r_{i,k}}/\mathbf{MSK}_k$ (where $r_{i,k} \in_R \mathbb{Z}_p$ and $\mathbf{MSK}_k \in_R \mathbb{H}$), and we combine them by computing

$$\mathbf{h}_{i,1}^{r_i}/\mathbf{MSK} = \prod_{\mathcal{A}_k} \mathbf{h}_{i,1}^{r_{i,k}}/\mathbf{MSK}_k = \mathbf{h}_{i,1}^{\sum_{\mathcal{A}_k} r_{i,k}} / \prod_{\mathcal{A}_k} \mathbf{MSK}_k. \tag{C.1}$$

Then we know that $r_i = \sum_{\mathcal{A}_k} r_{i,k}$ and $\mathbf{MSK} = \prod \mathbf{MSK}_k$ are secret as long as one of the authorities is honest (and all $\mathbf{h}_{i,1}^{r_{i,k}}/\mathbf{MSK}_k$ were well-formed, which they prove by using an instance of Okamoto's identification protocol (see Section 2.7.2)).

Now, we have to do this for all $i \in [1, n_1]$, because per assumption, we cannot generate $\mathbf{h}_{i',1}^{r_{i'}}/\mathbf{MSK}$ from $\mathbf{h}_{i,1}^{r_i}/\mathbf{MSK}$ for $i' \neq i$, because r_i and $\mathbf{h}_{i,1}^{r_i}$ are secret. Because we want to preserve the form of the secret keys, we define fixed generation parameters, such that $\mathbf{h}_{i',1}^{r_{i'}}/\mathbf{MSK}$ can be generated in secret, but the rest of the parameters can still be generated, i.e. we have fixed generation parameters:

$$\begin{aligned}
\mathbf{FGP}_{1,i} &= \mathbf{h}_0^{r_i} \\
\mathbf{FGP}_{2,i,j} &= \begin{cases} \mathbf{h}_{i,j}^{r_i}/\mathbf{h}_0^{v_j} & \text{if } j > 1 \\ \mathbf{h}_{i,1}^{r_i}/\mathbf{MSK} & \text{if } j = 1 \end{cases} \\
\mathbf{FGP}_{3,i,\ell,j} &= \mathbf{h}_{\ell,j}^{r_i} \\
\mathbf{FGP}_{4,i,y} &= \prod_{t \in [0,T]} (\mathbf{h}'_{i,t})^{r_i y^t} \\
\mathbf{FGP}_{5,i,\ell,t} &= (\mathbf{h}'_{\ell,t})^{r_i},
\end{aligned}$$

where r_i generated in a distributed fashion, i.e. each authority $\mathcal{A}_k$ picks random $\alpha_{1,k}, \alpha_{2,k}, r_{1,k}, ..., r_{n_1,k} \in_R \mathbb{Z}_p$ and computes:

$$\begin{aligned}
\mathbf{FGP}_{1,i,k} &= \mathbf{h}_0^{r_{i,k}} \\
\mathbf{FGP}_{2,i,j,k} &= \begin{cases} \mathbf{h}_{i,j}^{r_{i,k}} & \text{if } j > 1 \\ \mathbf{h}_{i,1}^{r_i}/(h^{\alpha_{1,k}}, h^{\alpha_{2,k}}) & \text{if } j = 1 \end{cases} \\
\mathbf{FGP}_{3,i,\ell,j,k} &= \mathbf{h}_{\ell,j}^{r_{i,k}} \\
\mathbf{FGP}_{4,i,y,k} &= \prod_{t \in [0,T]} (\mathbf{h}'_{i,t})^{r_{i,k} y^t} \\
\mathbf{FGP}_{5,i,\ell,t,k} &= (\mathbf{h}'_{\ell,t})^{r_{i,k}},
\end{aligned}$$

with Σ-proofs of EQ-compositions of Schnorr's and Okamoto's protocols to prove correctness of these shares, and then all individual shares are multiplied. Note that all of these values may be public, so the authorities can use a public bulletin board or other public communication channels in order to communicate these shares.

We can now rewrite the secret keys as

$$\begin{aligned}
\mathbf{SK}_{1,i} &= \mathbf{h}_0^{u_i} \cdot \mathbf{FGP}_{1,i} \\
\mathbf{SK}_{2,i,j} &= \begin{cases} \mathbf{h}_{i,j}^{u_i}/\mathbf{h}_0^{v_j} \cdot \mathbf{FGP}_{2,i,j} & \text{if } j > 1 \\ \mathbf{h}_{i,1}^{u_i} \cdot \mathbf{FGP}_{2,i,1} & \text{if } j = 1 \end{cases} \\
\mathbf{SK}_{3,i,\ell,j} &= \mathbf{h}_{\ell,j}^{u_i} \cdot \mathbf{FGP}_{3,i,\ell,j} \\
\mathbf{SK}_{4,i,y} &= \prod_{t \in [0,T]} (\mathbf{h}'_{i,t})^{u_i y^t} \cdot \mathbf{FGP}_{4,i,y} \\
\mathbf{SK}_{5,i,\ell,t} &= (\mathbf{h}'_{\ell,t})^{u_i} \cdot \mathbf{FGP}_{5,i,\ell,t},
\end{aligned}$$

where $u_i \in_R \mathbb{Z}_p$ for all $i \in [1, n_1]$.

Note that $\mathbf{FGP}_{4,i,y}$ is not public, but rather denotes part of the secret key of the authority that manages attribute y. Also note that $\mathbf{FGP}_{5,i,\ell,t}$ (and the shares generated by the authorities) cannot be computed in the fashion that we described, because it requires $\mathbf{h}'_{i,t}$ to be public for all i, t. At the end of the next section, we will discuss how we can securely generate both $\mathbf{FGP}_{4,i,y}$ and $\mathbf{FGP}_{5,i,\ell,t}$.

In order to be able to use our new formulation of the secret keys, we are supposed to show that it is equivalent to our old definition:

Proof. If we define our new formulation of the secret keys as

$$\begin{aligned}
\overline{\mathbf{SK}}_{1,i} &= \mathbf{h}_0^{u_i} \cdot \mathbf{FGP}_{1,i} \\
\overline{\mathbf{SK}}_{2,i,j} &= \begin{cases} \mathbf{h}_{i,j}^{u_i}/\mathbf{h}_0^{v_j} \cdot \mathbf{FGP}_{2,i,j} & \text{if } j > 1 \\ \mathbf{h}_{i,1}^{u_i} \cdot \mathbf{FGP}_{2,i,1} & \text{if } j = 1 \end{cases} \\
\overline{\mathbf{SK}}_{3,i,\ell,j} &= \mathbf{h}_{\ell,j}^{u_i} \cdot \mathbf{FGP}_{3,i,\ell,j} \\
\overline{\mathbf{SK}}_{4,i,y} &= \prod_{t \in [0,T]} (\mathbf{h}'_{i,t})^{u_i y^t} \cdot \mathbf{FGP}_{4,i,y} \\
\overline{\mathbf{SK}}_{5,i,\ell,t} &= (\mathbf{h}'_{\ell,t})^{u_i} \cdot \mathbf{FGP}_{5,i,\ell,t},
\end{aligned}$$

and we write our old keys as

$$\begin{aligned}
\mathbf{SK}_{1,i} &= \mathbf{h}_0^{r'_i} \\
\mathbf{SK}_{2,i,j} &= \begin{cases} \mathbf{h}_{i,j}^{r'_i}/\mathbf{h}_0^{v'_j} & \text{if } j > 1 \\ \mathbf{h}_{i,1}^{r'_i}/\mathbf{MSK} & \text{if } j = 1 \end{cases} \\
\mathbf{SK}_{3,i,\ell,j} &= \mathbf{h}_{\ell,j}^{r'_i} \\
\mathbf{SK}_{4,i,y} &= \prod_{t \in [0,T]} (\mathbf{h}'_{i,t})^{r'_i y^t} \\
\mathbf{SK}_{5,i,\ell,t} &= (\mathbf{h}'_{\ell,t})^{r'_i},
\end{aligned}$$

where $u_i, v_j, r'_i \in_R \mathbb{Z}_p$ and $v'_j = v_j$ for all i and j. Then $\overline{\mathbf{SK}}$ and $\mathbf{SK}$ are equivalent, because the distributions of $r'_i \in_R \mathbb{Z}_p$ and $r_i + u_i$, with $u_i \in_R \mathbb{Z}_p$ are equivalent for each i, and

$$\begin{aligned}
\overline{\mathbf{SK}}_{1,i} &= \mathbf{h}_0^{u_i} \cdot \mathbf{h}_0^{r_i} = \mathbf{h}_0^{u_i + r_i} \\
\overline{\mathbf{SK}}_{2,i,j} &= \begin{cases} \mathbf{h}_{i,j}^{u_i}/\mathbf{h}_0^{v_j} \cdot \mathbf{h}_{i,j}^{r_i} & \text{if } j > 1 \\ \mathbf{h}_{i,1}^{u_i} \cdot \mathbf{h}_{i,1}^{r_i}/\mathbf{MSK} & \text{if } j = 1 \end{cases} = \begin{cases} \mathbf{h}_{i,j}^{u_i + r_i}/\mathbf{h}_0^{v_j} & \text{if } j > 1 \\ \mathbf{h}_{i,1}^{u_i + r_i}/\mathbf{MSK} & \text{if } j = 1 \end{cases} \\
\overline{\mathbf{SK}}_{3,i,\ell,j} &= \mathbf{h}_{\ell,j}^{u_i} \cdot \mathbf{h}_{\ell,j}^{r_i} = \mathbf{h}_{\ell,j}^{u_i + r_i} \\
\overline{\mathbf{SK}}_{4,i,y} &= \prod_{t \in [0,T]} (\mathbf{h}'_{i,t})^{u_i y^t} \cdot \prod_{t \in [0,T]} (\mathbf{h}'_{i,t})^{r_i y^t} = \prod_{t \in [0,T]} (\mathbf{h}'_{i,t})^{(u_i + r_i) y^t} \\
\overline{\mathbf{SK}}_{5,i,\ell,t} &= (\mathbf{h}'_{\ell,t})^{u_i} \cdot (\mathbf{h}'_{\ell,t})^{r_i} = (\mathbf{h}'_{\ell,t})^{u_i + r_i}.
\end{aligned}$$

Hence, they are equivalent. $\square$

Note that in our new formulation, we have gotten rid of the master secret key **MSK**, which means that we have found a way to generate keys without using **MSK** in plain sight.

Finally, we note that $\mathbf{MSK}$ also occurs in the public parameters, i.e. $e(\mathbf{g}_0, \mathbf{MSK})$. We have

$$\begin{aligned} e(\mathbf{g}_0, \mathbf{MSK}) &= e(\mathbf{g}_0, \prod_{\mathcal{A}_k} \mathbf{MSK}_k) = \prod_{\mathcal{A}_k} e(\mathbf{g}_0, (h^{\alpha_{1,k}}, h^{\alpha_{2,k}})) \\ &= \prod_{\mathcal{A}_k} e(\mathbf{g}_0, (h,h))^{(\alpha_{1,k},\alpha_{2,k})} = \prod_{\mathcal{A}_k} \left(e(\phi_L(\mathbf{g}_0), h)^{\alpha_{1,k}} \cdot e(\phi_R(\mathbf{g}_0), h)^{\alpha_{2,k}}\right). \end{aligned}$$

Because $e(\phi_L(\mathbf{g}_0), h), e(\phi_R(\mathbf{g}_0), h) \in G_T$ are both public, the authorities can simply add $e(\phi_L(\mathbf{g}_0), h)^{\alpha_{1,k}}$ and $e(\phi_R(\mathbf{g}_0), h)^{\alpha_{2,k}}$ to the previously broadcast fixed generation parameters, together with a Σ-proof of equality of $\alpha_{1,k}$ and $\alpha_{2,k}$. Then $e(\mathbf{g}_0, \mathbf{MSK})$ can easily be generated.

C.2.2 Generating $\mathbf{SK}_{4,i,y}$

As we already mentioned, we can write $\mathbf{SK}_{4,i,y}$ as a power of two polynomials that are multiplied by some scalars and then added. We will now show that this is indeed the case, and how these polynomials relate to one another.

Recall that $\mathbf{h}_0 = h^{\mathbf{D}^*}$, for which $\mathbf{D}^*$ was defined as

$$\pi_L(\mathbf{B}^*\mathbf{R}) = \pi_L\left(\frac{1}{\det(\mathbf{B})}\begin{pmatrix} \mathbf{R}_{11}\mathbf{B}_{22} & -\mathbf{B}_{21} \\ -\mathbf{R}_{11}\mathbf{B}_{12} & \mathbf{B}_{11}\end{pmatrix}\right),$$

where $\mathbf{R}, \mathbf{B} \in \mathbb{Z}_p^{2\times 2}$, such that $\mathbf{B}$ is invertible and $\mathbf{R}$ is an invertible diagonal matrix with $\mathbf{R}_{22} = 1$. Because $\mathbf{R}_{11} \in_R \mathbb{Z}_p^*$, and $\det(\mathbf{B}) \neq 0$, we have $\frac{\mathbf{R}_{11}}{\det(\mathbf{B})} \in_R \mathbb{Z}_p^*$, which means we can write $\mathbf{h}_0$ as $(h^{x\mathbf{B}_{22}}, h^{-x\mathbf{B}_{12}})$.

For $\mathbf{h}'_{i,t}$, on the other hand, we have some $\mathbf{A}'_{i,t} \in_R \mathbb{Z}_p^{2\times 2}$ such that $\mathbf{h}'_{i,t} = h^{\mathbf{D}'^*_{i,t}}$ for which $\mathbf{D}'^*_{i,t}$ is defined as

$$\pi_L(\mathbf{B}^*(\mathbf{A}'_{i,t})^\intercal\mathbf{R}) = \frac{\mathbf{R}_{11}}{\det(\mathbf{B})}\begin{pmatrix} (\mathbf{A}'_{i,t})_{11}\mathbf{B}_{22} - (\mathbf{A}'_{i,t})_{12}\mathbf{B}_{21} \\ -(\mathbf{A}'_{i,t})_{11}\mathbf{B}_{12} + (\mathbf{A}'_{i,t})_{12}\mathbf{B}_{11}\end{pmatrix},$$

hence analogously to $\mathbf{h}_0$ we can write $\mathbf{h}'_{i,t}$ as

$$\begin{aligned} &\left(h^{x\left((\mathbf{A}'_{i,t})_{11}\mathbf{B}_{22} - (\mathbf{A}'_{i,t})_{12}\mathbf{B}_{21}\right)}, h^{x\left(-(\mathbf{A}'_{i,t})_{11}\mathbf{B}_{12} + (\mathbf{A}'_{i,t})_{12}\mathbf{B}_{11}\right)}\right) \\ &\qquad = \mathbf{h}_0^{(\mathbf{A}'_{i,t})_{11}} \cdot \left(h^{-x\mathbf{B}_{21}}, h^{x\mathbf{B}_{11}}\right)^{(\mathbf{A}'_{i,t})_{12}} \\ &\qquad \equiv \mathbf{h}_0^{(\mathbf{A}'_{i,t})_{11}} \cdot \bar{\mathbf{h}}_0^{(\mathbf{A}'_{i,t})_{12}}, \end{aligned}$$

for which we let $\bar{\mathbf{h}}_0 = \left(h^{x(a\mathbf{B}_{22} - b\mathbf{B}_{21})}, h^{x(-a\mathbf{B}_{12} + b\mathbf{B}_{11})}\right)$, with $a, b \in_R \mathbb{Z}_p$. Note that $\bar{\mathbf{h}}_0$ is of the same form as the other public parameters $\{\mathbf{h}_{i,j}, \mathbf{h}'_{i,t}\}_{i,j,t}$. Then we can write $\mathbf{SK}_{4,i,y}$ as

$$\left(\mathbf{h}_0^{\sum_{t\in[0,T]}(\mathbf{A}'_{i,t})_{11}\cdot y^t} \cdot \bar{\mathbf{h}}_0^{\sum_{t\in[0,T]}(\mathbf{A}'_{i,t})_{12}\cdot y^t}\right)^{r_i+u_i}, \tag{C.2}$$

which is indeed in the form that we discussed earlier, for polynomials $P_{1,i}(y) = \sum_{t\in[0,T]}(\mathbf{A}'_{i,t})_{11} \cdot y^t$ and $P_{2,i}(y) = \sum_{t\in[0,T]}(\mathbf{A}'_{i,t})_{12} \cdot y^t$.

So with inputs $\mathbf{SK}_{1,i}$ and $\bar{\mathbf{h}}_0^{r_i+u_i}$, each authority can compute $\mathbf{SK}_{4,i,y}$ for the attributes y that it manages.

Hence, with our decentralized VSS scheme in Definition 8, the authorities obtain their own shares of the polynomial $\mathrm{SK}_{\mathcal{A}_k,1,i} = P_{1,i}(y_k)$ and $\mathrm{SK}_{\mathcal{A}_k,2,i} = P_{2,i}(y_k)$, where y_k denotes an attribute that is managed by $\mathcal{A}_k$, such that they can compute $\mathbf{FGP}_{4,i,y_k}$ and eventually $\mathbf{SK}_{4,i,y}$. Note that in order to ensure that no group of corrupt authorities can retrieve the secret, we must assume that T is at least as large as the number of authorities. Also, whereas the rest of the communication may be public, the transfer of shares must be done in secret: each authority sends the share that is meant for another authority over a secure channel such that the particular share is only known to the sender and receiver.

Now, because $\mathbf{h}'_{i,t}$ are supposed to remain secret to all authorities under the assumption that at least one of the authorities is honest, we need to compute $\mathbf{FGP}_{5,i,\ell,t}$ from $\mathbf{FGP}_{1,i}$ and $\bar{\mathbf{h}}_0^{r_i}$. To this end, we introduce another fixed generation parameter $\mathbf{FGP}_{0,i} = \bar{\mathbf{h}}_0^{r_i}$ in the joint generation of $\mathbf{MSK}$ (with accompanying Σ-proof).

In order to reduce the number of exponentiations, we can also use the verification keys in the VSSS to compute $\mathbf{FGP}_{5,i,\ell,t}$ for all $t \in [0,T]$. Instead of computing the commitments as defined in Definition 8, we compute commitments $\mathbf{C}_{i,\ell,t} = (\mathbf{h}_0^{r_i})^{(\mathbf{A}'_{\ell,t})_{11}} \cdot (\bar{\mathbf{h}}_0^{r_i})^{(\mathbf{A}'_{\ell,t})_{12}}$ (each individual authority $\mathcal{A}_k$ computes 'subcommitments' $\mathbf{C}_{i,\ell,t,k} = (\mathbf{h}_0^{r_i})^{((\mathbf{A}'_{\ell,t})_{11})_k} \cdot (\bar{\mathbf{h}}_0^{r_i})^{((\mathbf{A}'_{i,t})_{12})_k}$ from which the commitments are computed). Then we can set $\mathbf{FGP}_{5,i,\ell,t} = \mathbf{C}_{i,\ell,t}$. Note that this way, we can only verify whether both polynomials are generated correctly, but if the verification step fails, we do not know which polynomial is incorrect. However, each authority can check whether the other authorities have generated their shares and subcommitments correctly, so if an authority was dishonest, we can detect this, even though we do not know where the authority has cheated.

C.3 Generating the rest of the parameters

Now that we have considered how we can generate $\mathbf{SK}_{2,i,1}$ and $\mathbf{SK}_{4,i,y}$ (and to some extent also $\mathbf{SK}_{5,i,\ell,t}$) and in addition to that the fixed generation parameters $\mathrm{FGP} = \{\mathbf{FGP}_{0,i}, \mathbf{FGP}_{1,i}, \mathbf{FGP}_{2,i,j}, \mathbf{FGP}_{3,i,\ell,j}, \mathbf{FGP}_{4,i,y}, \mathbf{FGP}_{5,i,\ell,t}\}_{i,\ell,j,y,t}$, such that $\mathbf{FGP}_{4,i,y}$ is only known to the authority that manages y and the rest is 'public'. However, we still have to consider the generation of $\mathbf{h}_0, \mathbf{h}_{i,j}$ and their 'counterparts' $\mathbf{g}_0, \mathbf{g}_{i,j}$, as well as the counterparts to $\mathbf{h}'_{i,t}$, namely $\mathbf{g}'_{i,t}$.

C.3.1 Generating $\mathbf{g}_0$ and $\mathbf{h}_0$

Recall that for $\mathbf{g}_0$ and $\mathbf{h}_0$ we have

$$\mathbf{g}_0 = g^{\mathbf{D}} = (g^{\mathbf{B}_{11}}, g^{\mathbf{B}_{21}}) \quad \mathbf{h}_0 = h^{\mathbf{D}^*} = (h^{x\mathbf{B}_{22}}, h^{-x\mathbf{B}_{12}}),$$

where $x \in_R \mathbb{Z}_p^*$ and $\mathbf{B}$ is an invertible matrix in $\mathbb{Z}_p^{2\times 2}$. A matrix is invertible if and only if its determinant is not equal to 0, i.e. we have $\mathbf{B}_{11}\mathbf{B}_{22} - \mathbf{B}_{12}\mathbf{B}_{21} \neq 0$. In other words, the authorities jointly generate $h^x = \prod_{\mathcal{A}_k} h^{x_k} \neq 1$ and then generate jointly

$$(g^{\mathbf{B}_{11}}, g^{\mathbf{B}_{21}}) = \prod_{\mathcal{A}_k} (g^{(\mathbf{B}_{11})_k}, g^{(\mathbf{B}_{21})_k})$$
$$(h^{x\mathbf{B}_{22}}, h^{-x\mathbf{B}_{12}}) = \prod_{\mathcal{A}_k} (h^{x(\mathbf{B}_{22})_k}, h^{-x(\mathbf{B}_{12})_k}).$$

Then they check whether the implicit $\mathbf{B}$ is invertible by computing

$$e(g^{\mathbf{B}_{11}}, h^{x\mathbf{B}_{22}}) \cdot e(g^{\mathbf{B}_{21}}, h^{-x\mathbf{B}_{12}})$$

and checking whether it is not equal to 1. If so, then we have found a suitable choice of $\mathbf{B}$, otherwise we repeat the process.

Alongside the generation of $\mathbf{g}_0$ and $\mathbf{h}_0$, the authorities also jointly generate $\bar{\mathbf{h}}_0 = \left(h^{x(a\mathbf{B}_{22}-b\mathbf{B}_{21})}, h^{x(-a\mathbf{B}_{12}+b\mathbf{B}_{11})}\right)$, and $\bar{\mathbf{g}}_0 = (g^{a\mathbf{B}_{11}+c\mathbf{B}_{12}}, g^{a\mathbf{B}_{21}+c\mathbf{B}_{22}})$, with $a, b, c \in_R \mathbb{Z}_p$. This is done in two steps: first g^a, g^c, h^a, h^b are jointly generated (with Σ-proofs of correctness), and then $\bar{\mathbf{g}}_0$ and $\bar{\mathbf{h}}_0$. Because the powers are equal to those of $\mathbf{g}_0$ and $\mathbf{h}_0$, the authorities use Σ-proofs of EQ-compositions of Schnorr's and Okamoto's protocol on their parts of the 'random exponentiation'.

C.3.2 Generating $\mathbf{g}_{i,j}$ and $\mathbf{h}_{i,j}$

For $\mathbf{g}_{i,j}$ and $\mathbf{h}_{i,j}$ we have

$$\begin{aligned}
\mathbf{g}_{i,j} &= g^{\mathbf{D}_{i,j}} = (g^{(\mathbf{A}_{i,j})_{11}\mathbf{B}_{11}+(\mathbf{A}_{i,j})_{21}\mathbf{B}_{12}}, g^{(\mathbf{A}_{i,j})_{11}\mathbf{B}_{21}+(\mathbf{A}_{i,j})_{21}\mathbf{B}_{22}}) \\
&\equiv \mathbf{g}_0^{(\mathbf{A}_{i,j})_{11}} \cdot \bar{\mathbf{g}}_0^{(\mathbf{A}_{i,j})_{21}} \\
\mathbf{h}_{i,j} &= h^{\mathbf{D}^*_{i,j}} = (h^{x((\mathbf{A}_{i,j})_{11}\mathbf{B}_{22}-(\mathbf{A}_{i,j})_{12}\mathbf{B}_{21}}, h^{-x((\mathbf{A}_{i,j})_{11}\mathbf{B}_{12}-(\mathbf{A}_{i,j})_{12}\mathbf{B}_{11}}) \\
&\equiv \mathbf{h}_0^{(\mathbf{A}_{i,j})_{11}} \cdot \bar{\mathbf{h}}_0^{(\mathbf{A}_{i,j})_{12}}.
\end{aligned}$$

So the authorities can easily generate these in a joint fashion as well. We note that we use EQ-compositions of Okamoto's protocol in the joint generation to ensure that $(\mathbf{A}_{i,j})_{11}$ in $\mathbf{g}_{i,j}$ and $\mathbf{h}_{i,j}$ are equal.

C.3.3 Generating $\mathbf{g}'_{i,t}$

Recall that in Section C.2.2, we had not explicitly defined $\mathbf{h}'_{i,t}$, but instead let the authorities jointly generate $\mathbf{FGP}_{5,i,\ell,t} = (\mathbf{h}'_{\ell,t})^{r_i}$. However, during this generation, the authorities generate 'subcommitments' $\mathbf{C}_{1,i,\ell,t,k} = (\mathbf{h}_0^{r_i})^{((\mathbf{A}'_{\ell,t})_{11})_k}$ and $\mathbf{C}_{2,i,\ell,t,k} = (\bar{\mathbf{h}}_0^{r_i})^{((\mathbf{A}'_{i,t})_{12})_k}$, which we can use in our generation of $\mathbf{g}'_{i,t}$:

$$\begin{aligned}
\mathbf{g}'_{i,t} &= g^{\mathbf{D}'_{i,t}} = (g^{(\mathbf{A}'_{i,t})_{11}\mathbf{B}_{11}+(\mathbf{A}'_{i,t})_{21}\mathbf{B}_{12}}, g^{(\mathbf{A}'_{i,t})_{11}\mathbf{B}_{21}+(\mathbf{A}'_{i,t})_{21}\mathbf{B}_{22}}) \\
&\equiv \mathbf{g}_0^{(\mathbf{A}'_{i,t})_{11}} \cdot \bar{\mathbf{g}}_0^{(\mathbf{A}'_{i,t})_{21}},
\end{aligned}$$

where $(\mathbf{A}'_{i,t})_{11}$ has to be equal to the $(\mathbf{A}'_{i,t})_{11}$ that occurs in $\mathbf{C}_{1,\ell,i,t} = (\mathbf{h}_0^{r_\ell})^{(\mathbf{A}'_{i,t})_{11}}$, and for this we can use an EQ-composition of an instance of Schnorr's and an instance of Okamoto's protocol on the 'subcommitments'.

C.3.4 Key generation parameters

In our new formulation of the set of secret keys, we compute the secret key from the fixed generation parameter and $\mathbf{h}_{\cdot}^{u_i}$, where $u_i \in_R \mathbb{Z}_p$. However, as we have seen in Section C.2.2, we cannot simply publish $\mathbf{h}'_{i,t}$, because then authorities can forge secret keys associated with attributes that they do not control. So instead, we apply the same tactics as in the creation of $\mathbf{SK}_{5,i,\ell,t}$: we publish $\mathbf{h}_{\ell,t}^{r'_i}$ for random $r'_i \in_R \mathbb{Z}_p$. Because the random variables in the exponent of all secret keys have to be the same, we do this for all parameters at the same time, with Σ-proofs to prove correctness.

To this end, we define key generation parameters KGP $= \{\mathbf{KGP}_{0,i}, \mathbf{KGP}_{1,i}, \mathbf{KGP}_{2,i,j}, \mathbf{KGP}_{3,i,\ell,j}, \mathbf{KGP}_{4,i,y}, \mathbf{KGP}_{5,i,\ell,t}\}$ such that

$$\begin{aligned}
\mathbf{KGP}_{0,i} &= \bar{\mathbf{h}}_0^{r'_i}\\
\mathbf{KGP}_{1,i} &= \mathbf{h}_0^{r'_i}\\
\mathbf{KGP}_{2,i,j} &= \mathbf{h}_{i,j}^{r'_i}\\
\mathbf{KGP}_{3,i,\ell,t} &= \mathbf{h}_{\ell,j}^{r'_i}\\
\mathbf{KGP}_{4,i,y_k} &= \prod_{t\in[0,T]} (\mathbf{h}'_{i,t})^{r'_i y_k^t} = \mathbf{KGP}_{1,i}^{\mathrm{SK}_{\mathcal{A}_k,1,i}} \mathbf{KGP}_{0,i}^{\mathrm{SK}_{\mathcal{A}_k,2,i}}\\
\mathbf{KGP}_{5,i,\ell,t} &= (\mathbf{h}'_{\ell,t})^{r'_i} = \mathbf{KGP}_{1,i}^{(\mathbf{A}'_{\ell,t})_{11}} \cdot \mathbf{KGP}_{0,i}^{(\mathbf{A}'_{\ell,t})_{12}}
\end{aligned}$$

are generated in a distributed fashion with $r'_i \in_R \mathbb{Z}_p$.

The new formulation of the secret keys will then be

$$\begin{aligned}
(\mathbf{SK}_{0,i} &= \mathbf{KGP}_{0,i}^{u_i} \cdot \mathbf{FGP}_{0,i})\\
\mathbf{SK}_{1,i} &= \mathbf{KGP}_{1,i}^{u_i} \cdot \mathbf{FGP}_{1,i}\\
\mathbf{SK}_{2,i,j} &= \begin{cases} \mathbf{KGP}_{2,i,j}^{u_i}/\mathbf{h}_0^{v_j} \cdot \mathbf{FGP}_{2,i,j} & \text{if } j > 1\\ \mathbf{KGP}_{2,i,1}^{u_i} \cdot \mathbf{FGP}_{2,i,1} & \text{if } j = 1 \end{cases}\\
\mathbf{SK}_{3,i,\ell,j} &= \mathbf{KGP}_{3,i,\ell,j}^{u_i} \cdot \mathbf{FGP}_{3,i,\ell,j}\\
\mathbf{SK}_{4,i,y} &= \mathbf{KGP}_{4,i,y}^{u_i} \cdot \mathbf{FGP}_{4,i,y}\\
\mathbf{SK}_{5,i,\ell,t} &= \mathbf{KGP}_{5,i,\ell,t}^{u_i} \cdot \mathbf{FGP}_{5,i,\ell,t},
\end{aligned}$$

which is quite obviously equivalent to the previous formulation of secret keys. Note that both $\mathbf{KGP}_{4,i,y}$ and $\mathbf{FGP}_{4,i,y}$ are only in the hands of authority $\mathcal{A}_k$ that manages attribute y.

Note that we can express $\mathbf{SK}_{4,i,y}$ in terms of $\mathbf{SK}_{0,i}, \mathbf{SK}_{1,i}$ and secret keys of authority $\mathcal{A}_k$ that manages y, i.e. $\mathrm{SK}_{\mathcal{A}_k,1,i}, \mathrm{SK}_{\mathcal{A}_k,2,i}$:

$$\begin{aligned}
\mathbf{SK}_{4,i,y} &= \mathbf{KGP}_{4,i,y}^{u_i} \cdot \mathbf{FGP}_{4,i,y}\\
&= \mathbf{KGP}_{1,i}^{\mathrm{SK}_{\mathcal{A}_k,1,i}} \mathbf{KGP}_{0,i}^{\mathrm{SK}_{\mathcal{A}_k,2,i}} \cdot \mathbf{FGP}_{1,i}^{\mathrm{SK}_{\mathcal{A}_k,1,i}} \mathbf{KGP}_{0,i}^{\mathrm{SK}_{\mathcal{A}_k,2,i}}\\
&= \mathbf{SK}_{1,i}^{\mathrm{SK}_{\mathcal{A}_k,1,i}} \mathbf{SK}_{0,i}^{\mathrm{SK}_{\mathcal{A}_k,2,i}}
\end{aligned}$$

C.3.5 Decentralized global and authority setup

We have discussed how we can generate the public parameters, so now we can apply our findings to the global setup, and also the authority setup, as it defines the secret keys associated with the authorities and the attributes they manage.

Towards this, we define a couple of algorithms:

- $\mathrm{GenGH_0}$: This algorithm generates $\mathbf{g}_0, \bar{\mathbf{g}}_0, \mathbf{h}_0$ and $\bar{\mathbf{h}}_0$ conform Section C.3.1.
- $\mathrm{GenGH_{i,j}}(\mathbf{g}_0, \mathbf{h}_0, \bar{\mathbf{g}}_0, \bar{\mathbf{h}}_0)$: On input $\mathbf{g}_0, \mathbf{h}_0, \bar{\mathbf{g}}_0, \bar{\mathbf{h}}_0$, this algorithm generates $\mathbf{g}_{i,j}$ and $\mathbf{h}_{i,j}$ for all i, j conform Section C.3.2.
- $\mathrm{GenFGP}(\mathrm{PPP})$: On input the partial public parameters PPP $= (\mathbf{g}_0, \mathbf{h}_0, \bar{\mathbf{g}}_0, \bar{\mathbf{h}}_0, \{\mathbf{g}_{i,j}, \mathbf{h}_{i,j}\}_{i,j})$, the algorithm generates the public parameter $e(\mathbf{g}_0, \mathbf{MSK})$ and

the first half of FGP (i.e. without $\mathbf{FGP}_{4,i,y}$ and $\mathbf{FGP}_{5,i,\ell,t}$) conform Section C.2.1.

- GenKGP(PPP): On input the partial parameters, this algorithm computes half of the key generation parameters KGP (without $\mathbf{KGP}_{4,i,y}$ and $\mathbf{KGP}_{5,i,\ell,t}$) conform Section C.3.4.
- GenSK($\mathbf{g}_0, \mathbf{h}_0, \bar{\mathbf{g}}_0, \bar{\mathbf{h}}_0$, FGP$_{01}$, KGP$_{01}$): On input parameters $\mathbf{g}_0, \mathbf{h}_0, \bar{\mathbf{g}}_0, \bar{\mathbf{h}}_0$, fixed generation parameters $\mathrm{FGP}_{01} = \{\mathbf{FGP}_{0,i}, \mathbf{FGP}_{1,i}\}_i$ and key generation parameters $\mathrm{KGP}_{01} = \{\mathbf{KGP}_{0,i}, \mathbf{KGP}_{1,i}\}_i$ this algorithm generates secret keys $\{\mathrm{SK}_{\mathcal{A}_k,1,i}, \mathrm{SK}_{\mathcal{A}_k,2,i}\}_i$ for each authority $\mathcal{A}_k$ conform Section C.2.2. As a side product, the algorithm also computes $\mathbf{FGP}_{5,i,\ell,t}$, $\mathbf{KGP}_{5,i,\ell,t}$ and $\mathbf{g}'_{i,t}$.

We use these algorithms to define our setup algorithms. The idea is that, after running the setups, we do not need any communication between the authorities. Because of the use of a master secret key **MSK** and its distributed generation, as well as the distributed generation of the authority-associated secret keys, this scheme is more similar to the MA-ABE scheme of Chase and Chow [CC09] than the decentralized scheme of Lewko and Waters [LW11]. However, our scheme tolerates the corruption of all but one authority ([CC09] assumes at least two honest authorities), which we will prove in Section C.5.

Another thing that is always defined in multi-authority schemes is the global identifier, GID, which is used to connect the keys to one user. Whereas this GID is usually one group element, we will use a slightly modified version that will fit our scheme better. Because any authority $\mathcal{A}_k$ that controls y_k can compute $\mathbf{SK}_{4,i,y_k}$ from $\mathbf{SK}_{1,i}$ and $\mathbf{SK}_{0,i}$, these make good choices for GID, i.e. we define GID $= \{\mathbf{SK}_{0,i}, \mathbf{SK}_{1,i}\}_{i\in[1,n_1]}$. This GID should be unique in the sense that for each pair of identifiers GID, $\overline{\mathrm{GID}}$, we have that $\mathbf{SK}_{0,i} \neq \overline{\mathbf{SK}}_{0,i}$ and $\mathbf{SK}_{1,i} \neq \overline{\mathbf{SK}}_{1,i}$ for all i.

C.4 The decentralized construction

We can use the previously defined algorithms to define our decentralized multi-authority ciphertext-policy attribute-based encryption (dMA-CP-ABE) scheme with constant-size ciphertexts. For this, we use the [AC16; CW14a] scheme that we defined in Definition 34. Note that we also consider an algorithm that does not exist in other MA-ABE schemes: UserSetup, that defines the GID and the rest of the parameters that are not generated by the authorities that manage attributes upon signing onto the system. In practice, a group of parameter generation authorities $\mathcal{PA}$ (that is at least as large as the group of attribute authorities) is supposed to do this, because the random integers u_i have to be unknown to the user (and this is the case under the assumption that at least one of the parameter generation authorities is honest).

Definition 35 (dMA-CP-ABE with short CT) *Let access structures be represented by LSSS matrices, denoted as $\mathbb{A} = (A, \rho)$, where A is a $n_1 \times n_2$ matrix with entries in $\mathbb{Z}_p$, where p is a prime of at least λ bits, and $\rho : [1, n_1] \to \mathcal{U}$, which maps the rows from A to the attributes, where $\mathcal{U} = \mathbb{Z}_p$ denotes the (large) universe of attributes. We denote A_i as the i-th row of A and $a_{i,j} = A_{ij}$ as the j-th element of the i-th row. For set of attributes $\mathcal{S} \subseteq \mathcal{U}$ we define $\Upsilon = \{i : i \in [1, n_1], \rho(i) \in \mathcal{S}\}$ to be the set of indices of rows in A associated with $\mathcal{S}$. Furthermore, for this scheme, we define $T \in \mathbb{N}$ to be the maximum number of attributes a user is allowed to possess, i.e. $|\mathcal{S}| \leq T$. Suppose that T is at least as large as the number of authorities. We denote an authority as $\mathcal{A}_k$, where y_k denotes the attribute that is managed by $\mathcal{A}_k$.*

(i) GlobalSetup(1^λ) : *The* GlobalSetup *algorithm takes the security parameter* λ *as input, and generates* (p, G, H, G_T, g, h)*, where* p *is a prime of at least* λ *bits, and* G *and* H *are cyclic groups of order* p*, with generators* $g \in G$ *and* $h \in H$*. We also define non-degenerate bilinear mapping* $e : G \times H \to G_T$*.*

The authorities jointly generate:

$$\begin{gathered}
\mathbf{g}_0, \bar{\mathbf{g}}_0, \mathbf{h}_0, \bar{\mathbf{h}}_0 \leftarrow \mathrm{GenGH}_0 \\
\{\mathbf{g}_{i,j}, \mathbf{h}_{i,j}\} \leftarrow \mathrm{GenGH}_{\mathrm{i,j}}(\mathbf{g}_0, \mathbf{h}_0, \bar{\mathbf{g}}_0, \bar{\mathbf{h}}_0) \\
\mathrm{PPP} \leftarrow (\mathbf{g}_0, \bar{\mathbf{g}}_0, \mathbf{h}_0, \bar{\mathbf{h}}_0, \{\mathbf{g}_{i,j}, \mathbf{h}_{i,j}\}) \\
(e(\mathbf{g}_0, \mathbf{MSK}), \{\mathbf{FGP}_{0,i}, \mathbf{FGP}_{1,i}, \mathbf{FGP}_{2,i,j}, \mathbf{FGP}_{3,i,\ell,j}\}) \leftarrow \mathrm{GenFGP}(\mathrm{PPP}) \\
\mathrm{FGP}' \leftarrow \{\mathbf{FGP}_{0,i}, \mathbf{FGP}_{1,i}, \mathbf{FGP}_{2,i,j}, \mathbf{FGP}_{3,i,\ell,j}\} \\
\mathrm{KGP}' := \{\mathbf{KGP}_{0,i}, \mathbf{KGP}_{1,i}, \mathbf{KGP}_{2,i,j}, \mathbf{KGP}_{3,i,\ell,j}\} \leftarrow \mathrm{GenKGP}(\mathrm{PPP}).
\end{gathered}$$

The authorities also generate the secret keys (implicitly) and the rest of the public parameters:

$$\begin{gathered}
\mathrm{FGP}_{01}, \mathrm{KGP}_{01} \leftarrow \{\mathbf{FGP}_{0,i}, \mathbf{FGP}_{1,i}\}_i, \{\mathbf{KGP}_{0,i}, \mathbf{KGP}_{1,i}\}_i \\
\{\mathbf{FGP}_{5,i,\ell,t}, \mathbf{KGP}_{5,i,\ell,t}, \mathbf{g}'_{i,t}\} \leftarrow \mathrm{GenSK}(\mathbf{g}_0, \mathbf{h}_0, \bar{\mathbf{g}}_0, \bar{\mathbf{h}}_0, \mathrm{FGP}_{01}, \mathrm{KGP}_{01}).
\end{gathered}$$

Let $\mathrm{FGP} = \mathrm{FGP}' \cup \{\mathbf{FGP}_{5,i,\ell,t}\}$ *and* $\mathrm{KGP} = \mathrm{KGP}' \cup \{\mathbf{KGP}_{5,i,\ell,t}\}$*. Then the global parameters are*

$$\mathrm{GP} = \Big(p, G, H, G_T, e, e(\mathbf{g}_0, \mathbf{MSK}), \mathbf{g}_0, \mathbf{h}_0, \{\mathbf{g}_{i,j}\}_{i,j}, \{\mathbf{g}'_{i,t}\}_{i,t}, \mathrm{FGP}, \mathrm{KGP}\Big).$$

(ii) AuthoritySetup(GP)*: Each authority* $\mathcal{A}_k$ *with attribute* y_k *has a set of pairs of secret keys,* $\{\mathrm{SK}_{\mathcal{A}_k,1,i}, \mathrm{SK}_{\mathcal{A}_k,2,i}\}_{i\in[1,n_1]}$*, that were defined during the global setup. The secret key associated with* $\mathcal{A}_k$ *is*

$$\mathrm{MSK}_{\mathcal{A}_k} = \{\mathrm{SK}_{\mathcal{A}_k,1,i}, \mathrm{SK}_{\mathcal{A}_k,2,i}\}_{i\in[1,n_1]}.$$

We define the public key of authority $\mathcal{A}_k$ *as the set of verification keys* $\mathrm{VK}_{\mathcal{A}_k,1} = \prod_{t\in[0,T]} \mathbf{C}_{1,2,t,k}^{y_k^t}$*, and* $\mathrm{VK}_{\mathcal{A}_k,i} = \prod_{t\in[0,T]} \mathbf{C}_{i,1,t,k}^{y_k^t}$ *for* $i \neq 1$ *such that*

$$\mathrm{PK}_{\mathcal{A}_k} = \{\mathrm{VK}_{\mathcal{A}_k,i}\}_{i\in[1,n_1]}.$$

(iii) UserSetup(GP)*: Upon signing onto the system, the user and the parameter generation authorities* $\mathcal{PA}$ *jointly generate*

$$\begin{aligned}
\mathbf{SK}_{0,i,\mathrm{ID}} &= \mathbf{KGP}_{0,i}^{u_i} \cdot \mathbf{FGP}_{0,i} \\
\mathbf{SK}_{1,i,\mathrm{ID}} &= \mathbf{KGP}_{1,i}^{u_i} \cdot \mathbf{FGP}_{1,i} \\
\mathbf{SK}_{2,i,j,\mathrm{ID}} &= \begin{cases} \mathbf{KGP}_{2,i,j}^{u_i} / \mathbf{h}_0^{v_j} \cdot \mathbf{FGP}_{2,i,j} & \text{if } j > 1 \\ \mathbf{KGP}_{2,i,1}^{u_i} \cdot \mathbf{FGP}_{2,i,1} & \text{if } j = 1 \end{cases} \\
\mathbf{SK}_{3,i,\ell,j,\mathrm{ID}} &= \mathbf{KGP}_{3,i,\ell,j}^{u_i} \cdot \mathbf{FGP}_{3,i,\ell,j} \\
\mathbf{SK}_{5,i,\ell,t,\mathrm{ID}} &= \mathbf{KGP}_{5,i,\ell,t}^{u_i} \cdot \mathbf{FGP}_{5,i,\ell,t},
\end{aligned}$$

such that $u_i, v_j, \in_R \mathbb{Z}_p$ *for all* $i \in [1, n_1]$ *and* $j \in [2, n_2]$ *and unknown under the assumption that at least one of the parameter generation authorities is honest. The global identifier is defined as* $\mathrm{GID} = \{\mathrm{ID}, \mathbf{SK}_{0,i,\mathrm{ID}}, \mathbf{SK}_{1,i,ID}\}_{i\in[1,n_1]}$*. Here,* ID *is a random identifier for the user that is purely used for notation purposes.*

(iv) $\mathrm{Encrypt}(\mathrm{GP}, (A, \rho), M)$: *The* Encrypt *algorithm takes the master public key as input, as well as the access structure* (A, ρ) *and message* $M \in G_T$*. Let* $s \in_R \mathbb{Z}_p$*. The output ciphertext is* $\mathrm{CT} = ((A, \rho), \mathbf{CT}_1, \mathbf{CT}_2, \mathrm{CT}_3)$*, such that*

$$\begin{aligned}
\mathbf{CT}_1 &= \mathbf{g}_0^s \\
\mathbf{CT}_2 &= \prod_{i \in [1,n_1], j \in [1,n_2]} \mathbf{g}_{i,j}^{s a_{i,j}} \prod_{i \in [1,n_1], t \in [0,T]} (\mathbf{g}'_{i,t})^{s \rho(i)^t} \\
\mathrm{CT}_3 &= M \cdot e(\mathbf{g}_0, \mathbf{MSK})^s.
\end{aligned}$$

(v) $\mathrm{KeyGen}(\mathrm{GID}, \mathcal{A}_k, y_k)$: *Authority* $\mathcal{A}_k$ *that manages attribute* y_k *generates a secret key for global identifier* $\mathrm{GID} = \{\mathbf{SK}_{0,i,\mathrm{ID}}, \mathbf{SK}_{1,i,\mathrm{ID}}\}_{i \in [1,n_1]}$ *by computing*

$$\forall i \in [1, n_1] : \quad \mathbf{SK}_{4,i,y_k,\mathrm{ID}} = \mathbf{SK}_{1,i,\mathrm{ID}}^{\mathrm{SK}_{\mathcal{A}_k,1,i}} \mathbf{SK}_{0,i}^{\mathrm{SK}_{\mathcal{A}_k,2,i}}.$$

The output is $\mathrm{SK}_{y_k,\mathrm{ID}} = \{\mathbf{SK}_{4,i,y_k,\mathrm{ID}}\}_{i \in [1,n_1]}$.

(vi) $\mathrm{Decrypt}(\mathrm{GP}, \mathrm{SK}, \mathrm{CT})$: *The* Decrypt *algorithm takes the master public key* MPK*, the secret key* $\mathrm{SK}_{\mathrm{ID}}$ *associated with user* GID *that possesses a set of attributes* $\mathcal{S}$ *and ciphertext* CT *encrypted under* (A, ρ) *as input, and then determines for the set of attributes* $\Upsilon \subseteq \mathcal{S}$ *associated with the rows of* A *the set* $\{\varepsilon_i\}_{i \in \Upsilon}$ *such that* $\sum_{i \in \Upsilon} \varepsilon_i A_i = (1, 0, ..., 0)$*. Then it computes* $C_2 = e(\mathbf{CT}_2, \prod_{i \in \Upsilon} \mathbf{SK}_{1,i,\mathrm{ID}}^{\varepsilon_i})$ *and for all* $i \in \Upsilon$*, it computes*

$$\begin{aligned}
\mathbf{S}_{2,i} &= \prod_{j \in [1,n_2]} \mathbf{SK}_{2,i,j,\mathrm{ID}}^{a_{i,j}}, & \mathbf{S}_{3,i} &= \prod_{\ell \in [1,n_1], \ell \neq i, j \in [1,n_2]} \mathbf{SK}_{3,i,\ell,j,\mathrm{ID}}^{a_{\ell,j}}, \\
\mathbf{S}_{4,i} &= \mathbf{SK}_{4,i,\rho(i),\mathrm{ID}}, & \mathbf{S}_{5,i} &= \prod_{\ell \in [1,n_1], \ell \neq i, t \in [0,T]} \mathbf{SK}_{5,i,\ell,t,\mathrm{ID}}^{\rho(\ell)^t},
\end{aligned}$$

then sets $C_1 = e\left(\mathbf{CT}_1, \prod_{i \in \Upsilon} (\mathbf{S}_{2,i}\mathbf{S}_{3,i}\mathbf{S}_{4,i}\mathbf{S}_{5,i})^{\varepsilon_i}\right)$*, and retrieves the plaintext by computing* $\mathrm{CT}_3 \cdot C_1 / C_2$.

Note that the encryption and decryption algorithms are identical to those of Definition 34, and therefore the scheme is also correct. We also observe that during the key generation, authority $\mathcal{A}_k$ can prove its honesty by producing Σ-proofs of EQ-compositions of Okamoto's protocol, such that it is proven that we have

$$\begin{aligned}
\forall i \in [1, n_1] : \quad & \mathbf{SK}_{4,i,y_k,\mathrm{ID}} = \mathbf{SK}_{1,i,\mathrm{ID}}^{\mathrm{SK}_{\mathcal{A}_k,1,i}} \mathbf{SK}_{0,i,\mathrm{ID}}^{\mathrm{SK}_{\mathcal{A}_k,2,i}} \\
\wedge \quad & \mathbf{VK}_{\mathcal{A}_k,i} = \mathbf{FGP}_{1,i}^{\mathrm{SK}_{\mathcal{A}_k,1,i}} \mathbf{FGP}_{0,i}^{\mathrm{SK}_{\mathcal{A}_k,2,i}}
\end{aligned}.$$

Remark: The decentralization of the scheme comes with a sacrifice: the old CP-ABE scheme in Definition 34 was a large universe construction. Because the authorities have to know which attributes they are going to manage in the setup, as they use this in the decentralized VSS scheme to distribute the shares of the secret keys, they cannot add any attributes later in the scheme without having to communicate with the other authorities. Another problem is that the addition of an attribute also requires that T is incremented, otherwise the authority with more secret keys has more information about the exponent of $\prod_t \mathbf{h}'_{i,t}$ than the other authorities, and perhaps a smaller subgroup of authorities can retrieve the secret, which is in contradiction with our assumption that the scheme allows corruption of all but one authority.

C.5 Security of the scheme

We will prove that our scheme is semi-adaptively secure against static corruption. We will do this by showing that the security game with our new formulation of the secret keys is computationally indistinguishable from the security game with secret keys conform Definition 34, because we already know that the scheme is proven secure in that security game. We will also show that the authorities have as much knowledge about the master secret key **MSK** and one another's secret keys as the users, which means that corruption of authorities does not lead to enough information to forge other secret keys. We will use the following 'properties':

- **Secret exponent:** In the joint generation of g^r by letting each authority $\mathcal{A}_k$ pick a random $r_k \in_R \mathbb{Z}_p$, and compute g^{r_k} together with a Σ-proof of knowledge of r_k, and setting $g^r = \prod_{\mathcal{A}_k} g^{r_k}$ such that $r = \sum_{\mathcal{A}_k} r_k$ holds implicitly, r is secret under the assumption that at least one of the authorities is honest.
- **Secret exponent II:** Under the SXDH assumption, we cannot compute $(h^{r'})^x$ from $h^r, h^{r'}$, and $(h^r)^x$ if r, r', x are unknown.
- **Security of honest authority's secret keys:** Because Shamir's secret sharing scheme is perfectly secure, no absolute subset of corrupt authorities $\mathcal{C}$ can compute $\mathbf{h}'_{i,t}$, the polynomials $P_{1,i}$ and $P_{2,i}$ as defined in Section C.2.2, or any honest authority's secret keys from $\mathrm{SK}_{\mathcal{A}_c,1,i}$, $\mathrm{SK}_{\mathcal{A}_c,2,i}$.
- **Preservation of DSG subgroup indistinguishability:** We have defined $\bar{\mathbf{h}}_0$ and $\bar{\mathbf{g}}_0$ such that they do not break left and right subgroup indistinguishability: they are both basically new public parameters of the same form as $\mathbf{g}_{i,j}$ and $\mathbf{h}_{i,j}$.

We had already shown that our new formulation of the secret keys (without $\mathbf{SK}_{0,i}$) is equivalent to our old formulation of our secret keys. With the preservation of DSG subgroup indistinguishability also follows that the addition of $\mathbf{SK}_{0,i}$ does not interfere with the security of the scheme, because it is of the same form as $\mathbf{h}_{i,j}$, and the addition of extra common variables to the scheme does not weaken the security of the scheme.

Because all parameters are generated from $\mathbf{g}_0$ and $\bar{\mathbf{g}}_0$, as well as $\mathbf{h}_0$ and $\bar{\mathbf{h}}_0$ by computing $\mathbf{h}_0^{r_1} \cdot \bar{\mathbf{h}}_0^{r_2}$ for random r_1 and r_2), all elements that are distributed are basically samples from $\mathbb{G}$ and $\mathbb{H}$ which are sampled with the sampling algorithms.

Lemma 36 *Our dMA-CP-ABE scheme with constant-size ciphertexts as defined in Definition 35 is semi-adaptively secure against static corruption in the standard model under the SXDH assumption.*

Proof sketch: We define the real game in the decentralized setup as the game with normal keys and ciphertexts as defined in Definition 35. We will prove that this game is indistinguishable from the security game with keys and ciphertexts as in Definition 34. Because the ciphertexts are exactly the same, we only have to transform the keys in the decentralized setting into keys in the centralized setting. Recall that we had already shown that our formulation of the secret keys is equivalent, e.g.:

$$\mathbf{SK}_{3,i,\ell,j} = \mathbf{KGP}^{u_i}_{3,i,\ell,j} \cdot \mathbf{FGP}_{3,i,\ell,j} = (\mathbf{h}^{r'_i}_{\ell,j})^{u_i} \cdot \mathbf{h}^{r_i}_{\ell,j} = \mathbf{h}^{r'_i u_i + r_i}_{\ell,j},$$

and because $u_i \in_R \mathbb{Z}_p$ can only be recovered when all parameter generation authorities are corrupt, and $r'_i, r_i \in_R \mathbb{Z}_p$ can only be recovered when all authorities work together, we have that $u_i r'_i + r_i \in_R \mathbb{Z}_p$ is unknown and uniformly distributed. (Note that the other secret keys are analogously defined.)

Moreover, corruption of authorities does not jeopardize the secret keys of honest authorities, because of the perfect secrecy of Shamir's SSS.

The distributed setup lets the authorities jointly generate all parameters by combining 'subparameters' that are created individually by the authorities. So in a sense, our instance of the dual system groups changes slightly here: if $\mathcal{A}$ is the set of authorities, then we can define our new (partial) set of public parameters as

$$\begin{aligned}&\{\mathbf{g}_0, \mathbf{h}_0, \bar{\mathbf{g}}_0, \bar{\mathbf{h}}_0, \{\mathbf{g}_{i,j}, \mathbf{g}'_{i,t}, \mathbf{h}_{i,j}, \mathbf{FGP}_{5,1,2,t}, \mathbf{FGP}_{5,i,1,t}\ (i \neq 1)\},\\ &\{\mathbf{g}_{i,j,k}, \mathbf{g}'_{i,t,k}, \mathbf{h}_{i,j,k}, \mathbf{FGP}_{5,1,2,t,k}, \mathbf{FGP}_{5,i,1,t,k}\ (i \neq 1)\}_{\mathcal{A}_k \in \mathcal{A}}\},\end{aligned}$$

for which easily follows that the DSG-properties hold (and as we had already shown, even left and right subgroup indistinguishability hold with the knowledge of all these extra parameters). Note that we have only included $\mathbf{FGP}_{5,1,2,t}$ and $\mathbf{FGP}_{5,i,1,t}$ because they represent $\mathbf{h}'_{i,t}$ for all i, t. The rest of the parameters that are used at any point in the setup can all be generated with the sampling algorithms. So the new scheme that follows from the DSG with these parameters is also semi-adaptively secure.

We have an extra property that we did not have in the first scheme: **MSK** can only be retrieved if all authorities work together, because of the secret exponent properties. So none of the authorities can decrypt ciphertexts for which they do not have authorization.

Hence, our scheme is indeed semi-adaptively secure against static corruption. $\square$

C.6 Efficiency

One of the other drawbacks of this scheme is that it is very inefficient. As was already the case in the centralized setup, the secret key generation costs a lot of computational power. Also, the setup itself generates more public parameters than the average CP-ABE scheme requires. This computational cost increases even further in this decentralized version, because each of the authorities generates every parameter in G once. The number of parameters in H blows up even more: each individual authority generates its own share of $\mathbf{h}_0, \bar{\mathbf{h}}_0, \{\mathbf{h}_{i,j}\}$ and all fixed and key generation parameters. The number of rounds of communication is not all that high, though, as there are a lot of parameters that do not depend on one another to be created. We need one round for $\mathbf{g}_0$ and $\mathbf{h}_0$, then one round for $\bar{\mathbf{g}}_0$ and $\bar{\mathbf{h}}_0$ (if we generate g^a, h^a etc. in the first round), the first part of the fixed and key generation parameters can be generated in parallel, and the secret keys and the corresponding fixed and key generation parameters can also be generated in parallel (but after the first part of FGP and KGP is generated). So in total we need four rounds of communication, with lots of computational cost: both for generation and verification, as each generation also comes with a number of Σ-proofs.

On the bright side, most of the computational power of the key generation algorithm is put on the side of the parameter generation authorities and user in this formulation of the scheme, which reduces the amount of computational power on the attribute authority's side. For each key request, the authority only has to compute $4n_1$ exponentations and $2n_1$ multiplications, which is much less than the original scheme needed. However, the computational power that is required from the parameter generation authorities is larger than in the central setup, as the security is only assured if the generation is distributed across multiple authorities (under the assumption that at least one of them is honest).

Appendix D

Correctness Proofs

D.1 Proof of correctness of the [AC16; CW14a] scheme

Proof. We have $\mathrm{CT}_3 = M \cdot e(\mathbf{g}_0, \mathbf{MSK})^s$ and dividing it by $e(\mathbf{g}_0, \mathbf{MSK})$ yields M. We indeed get $e(\mathbf{g}_0, \mathbf{MSK})$ by computing C_2/C_1, because we have

$$
\begin{aligned}
&\mathbf{C}_2/e\left(\mathbf{CT}_1, \prod_{i\in \mathrm{Y}} (\mathbf{S}_{2,i}\mathbf{S}_{3,i}\mathbf{S}_{4,i}\mathbf{S}_{5,i})^{\varepsilon_i}\right)\\
&= e\left(\prod_{i\in[1,n_1],j\in[1,n_2]} \mathbf{g}_{i,j}^{a_{i,j}} \prod_{i\in[1,n_1],t\in[0,T]} (\mathbf{g}'_{i,t})^{\rho(i)^t}, \prod_{i\in \mathrm{Y}} \mathbf{h}_{i,0}^{\varepsilon_i}\right) /\\
&e\left(\mathbf{g}_0, \prod_{i\in \mathrm{Y}}\left(\prod_{\ell\in[1,n_1],j\in[1,n_2]} \mathbf{h}_{i,\ell,j}^{a_{i,j}} \prod_{\ell\in[1,n_1],t\in[0,T]} (\mathbf{h}'_{i,\ell,t})^{\rho(\ell)^t} /\right.\right.\\
&\left.\left.\left(\mathbf{MSK}^{a_{i,1}} \prod_{j\in[2,n_2]} \mathbf{h}_{n_1+j-1,0}^{a_{i,j}}\right)\right)^{\varepsilon_i}\right)\\
&= e(g^{\mathbf{D}s}, \mathbf{MSK})\\
&e\left(\prod_{i\in[1,n_1],j\in[1,n_2]} (g^{\mathbf{D}_{i,j}s})^{a_{i,j}} \prod_{i\in[1,n_1],t\in[0,T]} (g^{\mathbf{D}'_{i,t}s})^{\rho(i)^t}, \prod_{i\in \mathrm{Y}} (h^{\mathbf{D}^* r_i})^{\varepsilon_i}\right) /\\
&e\left(g^{\mathbf{D}s}, \prod_{i\in \mathrm{Y}}\left(\prod_{\ell\in[1,n_1],j\in[1,n_2]} (h^{\mathbf{D}^*_{\ell,j} r_i})^{a_{\ell,j}} \prod_{\ell\in[1,n_1],t\in[0,T]} (h^{\mathbf{D}'^*_{\ell,t} r_i})^{\rho(\ell)^t}\right)^{\varepsilon_i}\right)\\
&= e(\mathbf{g}_0, \mathbf{MSK})^s \prod_{i\in \mathrm{Y}} \left(\frac{e(\prod_{\ell,j}(g^{\mathbf{D}_{\ell,j}s})^{a_{\ell,j}} \prod_{\ell,t}(g^{\mathbf{D}'_{\ell,t}s})^{\rho(\ell)^t}, h^{\mathbf{D}^* r_i})}{e(g^{\mathbf{D}s}, \prod_{\ell,j}(h^{\mathbf{D}^*_{\ell,j} r_i})^{a_{\ell,j}} \prod_{\ell,t}(h^{\mathbf{D}'^*_{\ell,t} r_i})^{\rho(\ell)^t}}\right)^{\varepsilon_i}\\
&= e(\mathbf{g}_0, \mathbf{MSK})^s \prod_{i\in \mathrm{Y}} \left(\prod_{\ell,j} \frac{e(g^{\mathbf{D}_{\ell,j}}, h^{\mathbf{D}^*})^{a_{\ell,j}}}{e(g^{\mathbf{D}}, h^{\mathbf{D}^*_{\ell,j}})^{a_{\ell,j}}} \prod_{\ell,t} \frac{e(g^{\mathbf{D}'_{\ell,t}\rho(\ell)^t}, h^{\mathbf{D}^*})}{e(g^{\mathbf{D}}, h^{\mathbf{D}'^*_{\ell,t}\rho(\ell)^t})}\right)^{r_i\varepsilon_i s}\\
&= e(\mathbf{g}_0, \mathbf{MSK})^s \prod_{i\in \mathrm{Y}} \left(\prod_{\ell,j} \frac{e(g,h)^{a_{\ell,j}\mathbf{D}^T_{\ell,j}\mathbf{D}^*}}{e(g,h)^{a_{\ell,j}\mathbf{D}^T\mathbf{D}^*_{\ell,j}}} \prod_{\ell,t} \frac{e(g,h)^{\rho(\ell)^t\mathbf{D}'^T_{\ell,t}\mathbf{D}^*}}{e(g,h)^{\rho(\ell)^t\mathbf{D}\mathbf{D}'^*_{\ell,t}}}\right)^{r_i\varepsilon_i s}\\
&= e(\mathbf{g}_0, \mathbf{MSK})^s \prod_{i\in \mathrm{Y}} \left(\prod_{\ell,j} 1_{\mathbb{G}_T} \prod_{\ell,t} 1_{\mathbb{G}_T}\right)^{r_i\varepsilon_i s} \quad = e(\mathbf{g}_0, \mathbf{MSK})^s,
\end{aligned}
$$

which follows from $\mathbf{D}^\intercal_{\ell,j}\mathbf{D}^* = \mathbf{D}^\intercal\mathbf{D}^*_{\ell,j}$ and $\mathbf{D}\mathbf{D}'^*_{\ell,t} = (\mathbf{D}'_{\ell,t})^\intercal\mathbf{D}^*$. Then, it follows from $(\mathbf{B}A_{\ell,j})^\intercal(\mathbf{B}^*\mathbf{R}) = A^\intercal_{\ell,j}\mathbf{B}^\intercal\mathbf{B}^*\mathbf{R} = \mathbf{A}^t_{\ell,j}{}^\intercal\mathbf{R}$ and $\mathbf{B}^\intercal\mathbf{B}^*\mathbf{A}^\intercal_{\ell,j}\mathbf{R} = \mathbf{A}^\intercal_{\ell,j}\mathbf{R}$). $\square$

List of Abbreviations

ABC	**A**ttribute-**B**ased **C**redentials
ABE	**A**ttribute-**B**ased **E**ncryption
CA	**C**ertificate **A**uthority or **C**entral **A**uthority
CCA	**C**hosen-**C**iphertext **A**ttack
CPA	**C**hosen-**P**laintext **A**ttack
CP-ABE	**C**iphertext-**P**olicy **A**trribute-**B**ased **E**ncryption
DBDH	**D**ecisional **B**ilinear **D**iffie-**H**ellman
DECODE	**Dec**entralized **C**itizen-**O**wned **D**ata **E**cosystems
DLIN	**D**ecisional **Lin**ear (assumption)
DSG	**D**ual **S**ystem **G**roup
G(B)GH	**G**eneric (**B**ilinear) **G**roup **H**euristic
GID	**G**lobal **Id**entifier
IBE	**I**dentity-**B**ased **E**ncryption
KGA	**K**ey **G**eneration **A**uthority
KP-ABE	**K**ey-**P**olicy **A**trribute-**B**ased **E**ncryption
LSI	**L**eft **S**ubgroup **I**nsdistinguishability
LSSS	**L**inear **S**ecret **S**haring **S**cheme
MA-ABE	**M**ulti-**A**uthority **A**ttribute-**B**ased **E**ncryption
MPC	**M**ulti**p**arty **C**omputation
MPK	**M**aster **P**ublic **K**ey
MSK	**M**aster **S**ecret/Private **K**ey
PES	**P**air **E**ncoding **S**cheme
PH	**P**arameter **H**iding
PK	**P**ublic **K**ey
PPT	**P**robabilistic **P**olynomial **T**ime
RSI	**R**ight **S**ubgroup **I**nsdistinguishability
SK	**S**ecret/Public **K**ey
SXDH	**S**ymmetric E**x**ternal **D**iffie-**H**ellman
TTP	**T**rusted **T**hird **P**arty

Bibliography

[AB+17] M. Al-Bassam, S. Bano, G. Danezis, M. deVilliers, and A. Sonnino. *Survey of Technlogies for ABC, Entitlements and Blockchains.* https://decodeproject.eu/publications/survey-technologies-abc-entitlements-and-blockchains. Accessed: 2018-02-20. 2017.

[AC16] S. Agrawal and M. Chase. "A Study of Pair Encodings: Predicate Encryption in Prime Order Groups". In: *Theory of Cryptography Conference.* Springer. 2016, pp. 259–288.

[AC17a] S. Agrawal and M. Chase. "FAME: Fast Attribute-Based Message Encryption". In: *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security.* ACM. 2017, pp. 665–682.

[AC17b] S. Agrawal and M. Chase. "Simplifying Design and Analysis of Complex Predicate Encryption Schemes". In: *Annual International Conference on the Theory and Applications of Cryptographic Techniques.* Springer. 2017, pp. 627–656.

[AI09] N. Attrapadung and H. Imai. "Attribute-Based Encryption Supporting Direct/Indirect Revocation Modes". In: *Ima International Conference on Cryptography and Coding.* Springer. 2009, pp. 278–300.

[Att+12] N. Attrapadung, J. Herranz, F. Laguillaumie, B. Libert, E. De Panafieu, and C. Ràfols. "Attribute-Based Encryption Schemes with Constant-Size Ciphertexts". In: *Theoretical Computer Science* 422 (2012), pp. 15–38.

[Att14] N. Attrapadung. "Dual System Encryption via Doubly Selective Security: Framework, Fully-Secure Functional Encryption for Regular Languages, and More". In: *Annual International Conference on the Theory and Applications of Cryptographic Techniques.* Springer. 2014, pp. 557–577.

[BBS04] D. Boneh, X. Boyen, and H. Shacham. "Short Group Signatures". In: *Annual International Cryptology Conference.* Springer. 2004, pp. 41–55.

[BBS98] M. Blaze, G. Bleumer, and M. Strauss. "Divertible Protocols and Atomic Proxy Cryptography". In: *International Conference on the Theory and Applications of Cryptographic Techniques.* Springer. 1998, pp. 127–144.

[Bei96] A. Beimel. "Secure Schemes for Secret Sharing and Key Distribution". PhD thesis. Ben Gurion University, 1996.

[Bel+09] M. Belenkiy, J. Camenisch, M. Chase, M. Kohlweiss, A. Lysyanskaya, and H. Shacham. "Randomizable Proofs and Delegatable Anonymous Credentials". In: *Advances in Cryptology-CRYPTO 2009.* Springer, 2009, pp. 108–125.

[BF01] D. Boneh and M. Franklin. "Efficient Generation of Shared RSA Keys". In: *Journal of the ACM (JACM)* 48.4 (2001), pp. 702–722.

[BSW07] J. Bethencourt, A. Sahai, and B. Waters. "Ciphertext-Policy Attribute-Based Encryption". In: *2007 IEEE Symposium on Security and Privacy (SP '07).* 2007, pp. 321–334.

[CC09] M. Chase and S. S. M. Chow. "Improving Privacy and Security in Multi-Authority Attribute-based Encryption". In: *Proceedings of the 16th ACM Conference on Computer and Communications Security*. CCS '09. ACM, 2009, pp. 121–130.

[CGW15] J. Chen, R. Gay, and H. Wee. "Improved Dual System ABE in Prime-Order Groups via Predicate Encodings". In: *Annual International Conference on the Theory and Applications of Cryptographic Techniques*. Springer. 2015, pp. 595–624.

[Cha07] M. Chase. "Multi-Authority Attribute-Based Encryption". In: *Theory of Cryptography Conference*. Springer. 2007, pp. 515–534.

[CHK03] R. Canetti, S. Halevi, and J. Katz. "A Forward-Secure Public-Key Encryption Scheme". In: *International Conference on the Theory and Applications of Cryptographic Techniques*. Springer. 2003, pp. 255–271.

[CN07] L. Cheung and C. Newport. "Provably Secure Ciphertext Policy ABE". In: *Proceedings of the 14th ACM conference on Computer and communications security*. ACM. 2007, pp. 456–465.

[Cra96] R. Cramer. "Modular Design of Secure Yet Practical Cryptographic Protocols". PhD thesis. 1996.

[CW14a] J. Chen and H. Wee. *Dual System Groups and its Applications – Compact HIBE and More*. IACR Cryptology ePrint Archive, Report 2014/265. https://eprint.iacr.org/2014/265.pdf. 2014.

[CW14b] J. Chen and H. Wee. "Semi-Adaptive Attribute-Based Encryption and Improved Delegation for Boolean Formula". In: *International Conference on Security and Cryptography for Networks*. Springer. 2014, pp. 277–297.

[CZF11] C. Chen, Z. Zhang, and D. Feng. "Efficient Ciphertext-Policy Attribute-Based Encryption with Constant-Size Ciphertext and Constant Computation-Cost". In: *International Conference on Provable Security*. Springer. 2011, pp. 84–101.

[Dan+17] G. Danezis, S. Bano, M. Al-Bassam, and A. Sonnin. *First Version of DECODE Architecture*. https://decodeproject.eu/publications/decode-architecture-first-version. Accessed: 2018-02-27. 2017.

[Den+14] H. Deng, Q. Wu, B. Qin, J. Domingo-Ferrer, L. Zhang, J. Liu, and W. Shi. "Ciphertext-Policy Hierarchical Attribute-Based Encryption with Short Ciphertexts". In: *Information Sciences* 275 (2014), pp. 370–384.

[DH76] W. Diffie and M. Hellman. "New Directions in Cryptography". In: *IEEE transactions on Information Theory* 22.6 (1976), pp. 644–654.

[DR13] J. Daemen and V. Rijmen. *The Design of Rijndael: AES – the Advanced Encryption Standard*. Springer Science & Business Media, 2013.

[ElG85] T. ElGamal. "A Public Key Cryptosystem and a Signature Scheme Based on Discrete Logarithms". In: *IEEE Transactions on Information Theory* 31.4 (1985), pp. 469–472.

[Emu+09] K. Emura, A. Miyaji, A. Nomura, K. Omote, and M. Soshi. "A Ciphertext-Policy Attribute-Based Encryption Scheme with Constant Ciphertext Length". In: *International Conference on Information Security Practice and Experience*. Springer. 2009, pp. 13–23.

[Fel87] P. Feldman. “A Practical Scheme for Non-Interactive Verifiable Secret Sharing”. In: *28th Annual Symposium on Foundations of Computer Science*. IEEE. 1987, pp. 427–438.

[Fre10] D. M. Freeman. “Converting Pairing-Based Cryptosystems from Composite-Order Groups to Prime-Order Groups”. In: *Annual International Conference on the Theory and Applications of Cryptographic Techniques*. Springer. 2010, pp. 44–61.

[FS86] A. Fiat and A. Shamir. “How to Prove Yourself: Practical solutions to Identification and Signature Problems”. In: *Conference on the Theory and Application of Cryptographic Techniques*. Springer. 1986, pp. 186–194.

[GH07] M. Green and S. Hohenberger. “Blind Identity-Based Encryption and Simulatable Oblivious Transfer”. In: *International Conference on the Theory and Application of Cryptology and Information Security*. Springer. 2007, pp. 265–282.

[Goy+06] V. Goyal, O. Pandey, A. Sahai, and B. Waters. “Attribute-based Encryption for Fine-grained Access Control of Encrypted Data”. In: *Proceedings of the 13th ACM Conference on Computer and Communications Security*. CCS ’06. ACM, 2006, pp. 89–98.

[Gui13] A. Guillevic. “Comparing the Pairing Efficiency over Composite-Order and Prime-Order Elliptic Curves”. In: *International Conference on Applied Cryptography and Network Security*. Springer. 2013, pp. 357–372.

[Han+12] J. Han, W. Susilo, Y. Mu, and J. Yan. “Privacy-Preserving Decentralized Key-Policy Attribute-Based Encryption”. In: *IEEE Transactions on Parallel and Distributed Systems* 23.11 (2012), pp. 2150–2162.

[Han+15] J. Han, W. Susilo, Y. Mu, J. Zhou, and M. H. Au. “Improving Privacy and Security in Decentralized Ciphertext-Policy Attribute-Based Encryption”. In: *IEEE Transactions on Information Forensics and Security* 10.3 (2015), pp. 665–678.

[HW14] S. Hohenberger and B. Waters. “Online/Offline Attribute-Based Encryption”. In: *International Workshop on Public Key Cryptography*. Springer. 2014, pp. 293–310.

[Jun+13] T. Jung, X. Y. Li, Z. Wan, and M. Wan. “Privacy Preserving Cloud Data Access With Multi-Authorities”. In: *INFOCOM, 2013 Proceedings IEEE*. IEEE. 2013, pp. 2625–2633.

[KSW08] J. Katz, A. Sahai, and B. Waters. “Predicate Encryption Supporting Disjunctions, Polynomial Equations, and Inner Products”. In: *Annual International Conference on the Theory and Applications of Cryptographic Techniques*. Springer. 2008, pp. 146–162.

[Lai+13] J. Lai, R. H. Deng, C. Guan, and J. Weng. “Attribute-Based Encryption with Verifiable Outsourced Decryption”. In: *IEEE Transactions on Information Forensics and Security* 8.8 (2013), pp. 1343–1354.

[Lew+10] A. Lewko, T. Okamoto, A. Sahai, K. Takashima, and B. Waters. “Fully Secure Functional Encryption: Attribute-Based Encryption and (Hierarchical) Inner Product Encryption”. In: *Annual International Conference on the Theory and Applications of Cryptographic Techniques*. Springer. 2010, pp. 62–91.

[Lew11] A. Lewko. *Decentralizing Attribute-Based Encryption*. 2011. URL: https://www.youtube.com/watch?v=vQvLTnoY9Nw (visited on 03/28/2018).

[Li+13] M. Li, S. Yu, Y. Zheng, K. Ren, and W. Lou. “Scalable and Secure Sharing of Personal Health Records in Cloud Computing using Attribute-Based Encryption”. In: *IEEE transactions on parallel and distributed systems* 24.1 (2013), pp. 131–143.

[Liu+11] Z. Liu, Z. Cao, Q. Huang, D. S. Wong, and T. H. Yuen. “Fully Secure Multi-Authority Ciphertext-Policy Attribute-Based Encryption Without Random Oracles”. In: *European Symposium on Research in Computer Security*. Springer. 2011, pp. 278–297.

[LW10] A. Lewko and B. Waters. *Decentralizing Attribute-Based Encryption*. Cryptology ePrint Archive, Report 2010/351. https://eprint.iacr.org/2010/351. 2010.

[LW11] A. Lewko and B. Waters. “Decentralizing Attribute-Based Encryption”. In: *Annual International Conference on the Theory and Applications of Cryptographic Techniques*. Springer. 2011, pp. 568–588.

[NYO08] T. Nishide, K. Yoneyama, and K. Ohta. “Attribute-Based Encryption with Partially Hidden Encryptor-Specified Access Structures”. In: *International Conference on Applied Cryptography and Network Security*. Springer. 2008, pp. 111–129.

[Oka92] T. Okamoto. “Provably Secure and Practical Identification Schemes and Corresponding Signature Schemes”. In: *Annual International Cryptology Conference*. Springer. 1992, pp. 31–53.

[OSW07] R. Ostrovsky, A. Sahai, and B. Waters. “Attribute-Based Encryption with Non-Monotonic Access Structures”. In: *Proceedings of the 14th ACM conference on Computer and communications security*. ACM. 2007, pp. 195–203.

[Ped91] T. P. Pedersen. “A Threshold Cryptosystem Without a Trusted Party”. In: *Workshop on the Theory and Application of of Cryptographic Techniques*. Springer. 1991, pp. 522–526.

[Pir+10] M. Pirretti, P. Traynor, P. McDaniel, and B. Waters. “Secure Attribute-Based Systems”. In: *Journal of Computer Security* 18.5 (2010), pp. 799–837.

[Qia+15] H. Qian, J. Li, Y. Zhang, and J. Han. “Privacy-Preserving Personal Health Record Using Multi-Authority Attribute-Based Encryption with Revocation”. In: *International Journal of Information Security* 14.6 (2015), pp. 487–497.

[QLZ13] H. Qian, J. Li, and Y. Zhang. “Privacy-Preserving Decentralized Ciphertext-Policy Attribute-Based Encryption with Fully Hidden Access Structure”. In: *International Conference on Information and Communications Security*. Springer. 2013, pp. 363–372.

[RD13] Y. Sreenivasa Rao and R. Dutta. “Decentralized Ciphertext-Policy Attribute-Based Encryption Scheme with Fast Decryption”. In: *IFIP International Conference on Communications and Multimedia Security*. Springer. 2013, pp. 66–81.

[RS91] C. Rackoff and D. R. Simon. "Non-Interactive Zero-Knowledge Proof of Knowledge and Chosen Ciphertext Attack". In: *Annual International Cryptology Conference*. Springer. 1991, pp. 433–444.

[RSA78] R. L. Rivest, A. Shamir, and L. Adleman. "A Method for Obtaining Digital Signatures and Public-Key Cryptosystems". In: *Communications of the ACM* 21.2 (1978), pp. 120–126.

[RW13] Y. Rouselakis and B. Waters. "Practical Constructions and New Proof Methods for Large Universe Attribute-Based Encryption". In: *Proceedings of the 2013 ACM SIGSAC Conference on Computer & communications Security*. ACM. 2013, pp. 463–474.

[RW15] Y. Rouselakis and B. Waters. "Efficient Statically-Secure Large-Universe Multi-Authority Attribute-Based Encryption". In: *International Conference on Financial Cryptography and Data Security*. Springer. 2015, pp. 315–332.

[Sah99] A. Sahai. "Non-Malleable Non-Interactive Zero Knowledge and Adaptive Chosen-Ciphertext Security". In: *Foundations of Computer Science, 1999. 40th Annual Symposium on*. IEEE. 1999, pp. 543–553.

[Sch17] B. Schoenmakers. *Cryptographic Protocols*. Lecture Notes. 2017.

[Sch91] C. P. Schnorr. "Efficient Signature Generation by Smart Cards". In: *Journal of Cryptology* 4.3 (1991), pp. 161–174.

[Sha79] A. Shamir. "How to Share a Secret". In: *Communications of the ACM* 22.11 (1979), pp. 612–613.

[Sha84] A. Shamir. "Identity-Based Cryptosystems and Signature Schemes". In: *Workshop on the Theory and Application of Cryptographic Techniques*. Springer. 1984, pp. 47–53.

[Shi+15] Y. Shi, Q. Zheng, J. Liu, and Z. Han. "Directly Revocable Key-Policy Attribute-Based Encryption with Verifiable Ciphertext Delegation". In: *Information Sciences* 295 (2015), pp. 221–231.

[SSW12] A. Sahai, H. Seyalioglu, and B. Waters. "Dynamic Credentials and Ciphertext Delegation for Attribute-Based Encryption". In: *Advances in Cryptology–CRYPTO 2012*. Springer, 2012, pp. 199–217.

[SW05] A. Sahai and B. Waters. "Fuzzy Identity-Based Encryption". In: *Annual International Conference on the Theory and Applications of Cryptographic Techniques*. Springer. 2005, pp. 457–473.

[Wat09] B. Waters. "Dual System Encryption: Realizing Fully Secure IBE and HIBE under Simple Assumptions". In: *Advances in Cryptology–CRYPTO 2009*. Springer, 2009, pp. 619–636.

[Wat11] B. Waters. "Ciphertext-Policy Attribute-Based Encryption - An Expressive, Efficient, and Provably Secure Realization". In: *International Workshop on Public Key Cryptography*. Springer. 2011, pp. 53–70.

[Yam+11] S. Yamada, N. Attrapadung, G. Hanaoka, and N. Kunihiro. "Generic Constructions for Chosen-Ciphertext Secure Attribute Based Encryption". In: *International Workshop on Public Key Cryptography*. Springer. 2011, pp. 71–89.

[Yu+10] S. Yu, C. Wang, K. Ren, and W. Lou. “Attribute-Based Data Sharing with Attribute Revocation”. In: *Proceedings of the 5th ACM Symposium on Information, Computer and Communications Security*. ACM. 2010, pp. 261–270.

[Zho+18] H. Zhong, W. Zhu, Y. Xu, and J. Cui. “Multi-Authority Attribute-Based Encryption Access Control Scheme with Policy Hidden for cloud Storage”. In: *Soft Computing* 22.1 (2018), pp. 243–251.