Anonymous Access Control with Attribute-Based Encryption

Robert R. Enderlein

School of Computer and Communication Sciences

Master's Thesis

March 18, 2011

EPFL Supervisor
Prof. Serge Vaudenay
EPFL / LASEC

IBM Supervisor
Dr. Jan Camenisch
IBM Zürich Research Lab

This thesis is submitted in partial fulfilment of the requirements for the degree of Master of Science in Communication Systems at the École Polytechnique Fédérale de Lausanne (EPFL). The thesis was written during a six-month project from September 21st, 2010 to March 18, 2011 at the IBM Zurich Research Laboratory in Rüschlikon under the supervision of Dr. Jan Camenisch.

Acknowledgements: *I thank Dr. Jan Camenisch, Maria Dubovitskaya, and Dr. Gregory Neven for their supervision, advice, comments, and for proofreading my report ; Prof. Serge Vaudenay for accepting to supervise my master's thesis ; Marisa Marciano Wynn, and Sylviane Dal Mas for helping me find this position. I thank the IBM Zurich Research Laboratory in Rüschlikon for financial support, and all its members for making my stay a pleasant one. Last but not least, I am grateful to my parents for support, and for doing my laundry and nice meals during the week-ends :-) .*

Abstract: *We present a scheme that allows secure and privacy-preserving access control of stored data — e.g., medical records — on potentially untrusted databases.*

Each record is protected by a hidden access-control policy. Users receive their keys from a potentially untrusted issuer. Records can be accessed fully anonymously: the databases never learn anything about the requester nor his choice. An extension to our scheme allows for expiration and revocation of records and keys.

Our solution combines hidden–ciphertext-policy attributed-based encryption with oblivious transfer. It is proven secure in the CRS model (without random oracles) under the generic bilinear group setting and SXDH assumption. Our scheme is more efficient and allows a larger class of access control policies than existing approaches based on credentials. We have fully implemented it in C++. It is scalable and fast enough to be used in practical settings.

Keywords: *Privacy, Oblivious Transfer, Attribute-Based Encryption.*

Contents

1 Introduction

1.1 Motivation

Existing cryptographic techniques do not provide sufficient privacy guarantees to handle long-term storage of medical data. Conventional key management techniques fail to take into account that at record creation, the set of users which are authorized to decrypt are not known by name—in fact the doctor you will consult in eight years time might not have entered university yet. It would be much more preferable to be able to specify the recipients by role, such as “all practicing cardiologists”, and trust the key issuing authority to manage the set of permissions each doctor has. For the sake of patient privacy, the set of authorized recipients of a record must remain hidden: not many people would like it to be known that they are being treated by an oncologist, a psychiatrist, or a plastic surgeon. The doctor in turn would like not to be bothered by the press if he is treating a celebrity patient.

Medical data is subject to a set of stringent laws. In Switzerland for example, it must be kept for 10 years and then destroyed. This means that there must exist some mechanism enforcing expiration/revocation of the records (and of course, sane key-management also requires expiration/revocation of the doctors’ keys). Each decryption attempt must therefore be vetted by the database maintainer. Of course, in order to safeguard the privacy of the patients, this revocation check must not reveal anything besides whether the record and key are fresh.

Our scheme could also be useful in a military setting: documents are generally assigned a classification (“secret”, “top secret”, etc.) and not encrypted to specific users. Is often desirable to hide the classification level of a document to prevent targeted attacks. It is useful to hide the set of users who have access to a specific file to prevent bribery or social engineering attacks. Attempts to decrypt the whole database by a malicious user, Wikileaks style, can de detected at the decryption servers and curtailed. Expiration and revocation of keys are also paramount in this setting.

1.2 Ingredients

In this paper, we present a scheme that combines hidden–ciphertext-policy attribute-based encryption (HABE) [NYO08, Nis08], adaptive oblivious transfer (OT) [CNS07], and (optionally) anonymous revocation [NFHF09, CDNZ11].

In Nishide et al.’s construction [NYO08] a trusted issuer selects a number of categories, and for each category chooses a set of permissible attributes. Each user receives a key associated with a list of attributes: one from each category. Each ciphertext is associated with an access structure (or policy), which is a subset of attributes. This access structure is hidden, meaning that one cannot recover it from the ciphertext alone. A user can decrypt a ciphertext if each attribute in his key is an element of the ciphertext access structure. One could view the ciphertext policy as implementing a limited version of conjunctive normal form: within a category, the policy specifies an OR condition on the attribute the user key has for that category; and all attributes of the key have to be in the access structure, basically an AND condition.

1.3 Our Contribution

We modify Nishide et al.’s scheme so that: firstly, users cannot decrypt offline anymore (so that the database retains control over the ciphertext, and can thus limit the number of decryption queries or enforce expiration/revocation); secondly, encryptors prove their ciphertext was computed correctly; and finally, the issuer doesn’t have to be a trusted party.

Roughly speaking, our scheme works like this: the issuer defines a special zeroth category containing only one permissible attribute, he labels the latter "issuer". All keys he issues will therefore contain this "issuer" attribute. Each database in the system define a new permissible attribute for that zeroth category, which it labels "database ϱ" (ϱ being the database index). When a database issues a record, it excludes the "issuer" attribute from the access structure, but includes its own "database ϱ" attribute. A user cannot decrypt on his own, since the "issuer" attribute of his key is not in the access structure of the ciphertext. To decrypt, he needs to query the database for a piece of information which will allow him to proceed with the decryption of the ciphertext without worrying about the zeroth category. This piece of information is tailor-made for his key and his choice of record, so he cannot use it to decrypt another record, and none of his friends can use it to decrypt the same record with their keys.

To make sure the query protocol reveals no information to the database, and at the same time make sure the user doesn't try to pass invalid ciphertexts to the database, parts of each record and parts of each key are signed with a signature scheme that allows zero-knowledge proof-of-possession. For this purpose we make heavy use of zero-knowledge proofs [CDM00] and a signature scheme that works with elements of a bilinear group [AFG$^+$10, AHO10]. This idea is somewhat similar to the concept of k-out-of-n oblivious transfer [CNS07].

To handle the case where the database might be dishonest, we require that all databases publish a non-interactive proof [GS08] that the ciphertexts they issue were computed honestly.[1] We need that proof to be non-interactive so that the record issuing protocol runs in constant time with respect to the number of users.[2]

To handle the case where the issuer and the database both are dishonest, we transform the key issuing algorithm of Nishide et al.'s scheme [NYO08] into an interactive protocol between the issuer and the user. Without this step, an honest user who is not authorized to decrypt a ciphertext might nevertheless recover the correct plaintext due to the way the dishonest issuer and database chose the supposedly random values in his key and ciphertext.[3]

Our construction is secure under the generic bilinear group model with type-3 pairings [GPS08].

1.4 Related Work

Attribute-Based Encryption There are a number of papers on the subject of attribute-based encryption [SW05, BSW07, NYO08, LOS$^+$10] and the related concept of predicate encryption [KSW08, LOS$^+$10]. All of these schemes of course allow offline decryption, which basically means that expiration/revocation mechanisms are impractical or insecure.

Some of these papers [KSW08, NYO08, LOS$^+$10] are *attribute-hiding*, meaning that users cannot deduce anything about the policy of a ciphertext except the fact that their key satisfies it or not. Katz et. al's [KSW08] and Lewko et. al's [LOS$^+$10] schemes allows for conjunctions and disjunctions in the policy, but requires composite-order bilinear groups (order is the product of three primes). Nishide, Yoneyama and Otha's scheme [NYO08] allows for slightly less powerful policies (*multi-valued attributes*) as described in Section 2.2.

All of these schemes except for Lewko et al.'s [LOS$^+$10] and the second construction of

[1] Users who try and decrypt a dishonest ciphertext will each recover a different plaintext, even if their key satisfies the policy of the ciphertext.

[2]We make the assumption that the database is not directly involved with broadcasting the ciphertext to all users. It can for example publish it on a web server, an anonymous ftp server, or ship a DVD containing all ciphertexts to all users.

[3]This problem is arguably only of limited interest in the real world, since the only harm caused to the user in that case is that he deduces incorrect information about the policy of the record. His privacy is maintained. However, it is necessary to handle this step to ensure that our security proof is sound.

Nishide et al. [NYO08][4] are only *selectively* secure, meaning that the adversary must commit to the policies he wishes to be challenged on *before he sees the system public key.* The security proof of Nishide et al.'s second construction is in the generic group model [Nis08]. Lewko et al.[LOS$^+$10] claimed they introduced the first fully secure ABE scheme, but they use composite-order bilinear groups, which cannot be generated without a trusted third party.

Oblivious transfer There is an extensive body of literature on the subject of oblivious transfer (OT). The variant of OT that we use in this paper is the k-out-of-n variant, which was introduced by Naor and Pinkas [NP99]. In k-out-of-n OT, a user may query for up to k different ciphertexts from a database of n ciphertexts. The database is guaranteed that the user didn't learn anything about the $(n-k)$ records that weren't queried. The user is guaranteed that the database didn't learn anything about his choice, and that the database didn't substitute any record with another one between two of his queries. Camenisch et al. [CNS07] introduced the adaptive variant of k-out-of-n OT in 2007, and gave an efficient constructions in the standard and random oracle models.

Several additions to the adaptive k-out-of-n OT protocol were made following the publication of the original paper. For instance, Camenisch et al. [CDN09] introduced access control in 2009: users are issued credentials, and a list of required credentials is associated to each record. To complete the OT protocol, the user must prove he possesses all credentials associated to the ciphertext he wants to access. Compared to our scheme however, the list of required credentials is in the clear (even though the list is not revealed during the query protocol).

The recent paper by Camenisch et al. [CDNZ11] is perhaps the most similar to our approach. Instead of attribute-based-encryption, their approach was to use credentials to enforce access control. Their ciphertext policy is slightly weaker than ours: in their scheme they associate a set of attributes (from a finite universe of ℓ attributes) to every credential and ciphertext; decryption is possible if and only if all attributes in the ciphertext are present in the credential. Our ciphertext-policy subsumes theirs.[5] Furthermore, the communication cost and the computational cost borne by the database in each query is linear to the number of attributes in their scheme, versus constant in our scheme (the price to pay is that in our scheme, the size of the user keys is larger by a much bigger constant factor than theirs)—our scheme is therefore a better choice if a large number of queries are to be expected (as soon as users query asymptotically more than a constant number of ciphertexts).

Nakanishi et al. [NFHF09] introduced revocable group signatures with constant-time and zero-knowledge verification, which can be used to implement revocation of user keys [CDNZ11]. We also use their ideas in our scheme for the same purpose.

It is of course possible to implement an oblivious transfer protocol with hidden access policy of arbitrary complexity by using Yao's generic two-party computation techniques [Yao82]; however the communication cost associated with each query will be linear in the size of the database times the number of users, versus constant-time[6] in our approach.

[4] It is not clear from their conference paper [NYO08] that their second construction is *non-selectively* secure. This statement (together with the proof of security of the scheme) appears in Nishide's PhD thesis [Nis08, p. 23].

[5]We set the same number of categories n in our scheme than they have attributes ℓ: $n = \ell$. Each category in our scheme will have two attributes: 0 and 1. When they issue a key with credentials $\{d_i\}_{i=1}^{\ell}$, we issue a key with attribute L_i such that $\forall i \in \mathbb{N}_{n+1}^* : L_i = d_i$. When they issue a record with ACL $\{c_i\}_{i=1}^{\ell}$, we issue a record with ciphertext policy $W_i = \{0, 1\}$ if $c_i = 0$, and with policy $W_i = \{1\}$ if $c_i = 1$.

[6]Assuming a constant security parameter κ. If the security parameters is not constant, then the run time is $O(\kappa^3)$.

2 Definitions

An anonymous access control with Hidden–ciphertext-policy attributed-based encryption and oblivious transfer (HABE-OT) scheme is run between the following parties:

- **An issuer**, who is responsible for setting up the system, generating the keys of the users, and revoking them.
- **Database maintainers** (databases for short), who are responsible for encrypting the records—which they can then publish on a regular file-server or ship on a DVD to all users [CNS07]—, revoking the records' ciphertexts, and assisting the users during the decryption protocol. Several independent databases may coexist in the system.
- **Users**, who want to decrypt ciphertexts, and must interact with the database in order to do so. After they received their key, users are completely anonymous and the choice of ciphertexts they submit for decryption will remain private.

Let $\mathcal{I}$ denote the issuer, $\mathcal{D}$ the set of all databases, $\mathcal{D}_\varrho$ the database with index ϱ, $\mathcal{U}$ the set of all users, and $\mathcal{U}_\varphi$ the user with index φ.

2.1 Syntax

By $\mathbb{N}$ we denote the set of natural numbers, by $\mathbb{N}_n$ the set of all natural numbers between 0 and $(n-1)$. By $\mathbb{Z}_p$ we denote the ring of integers modulo p. By $\mathbb{G}_1$, $\mathbb{G}_2$, $\mathbb{G}_\mathrm{T}$ we denote bilinear groups of prime order p. We use $\mathbb{N}_n^*$ etc. to denote $\mathbb{N}_n \setminus \{0\}$.

If $\kappa \in \mathbb{N}$, then $\mathbf{1}^\kappa$ denotes the string consisting of κ ones. If $\mathbb{A}$ is a set, then $a \overset{\$}{\leftarrow} \mathbb{A}$ means we set a to a random element of that set. If $\mathcal{A}$ is a Probabilistic Polynomial-Time (PPT) algorithm or interactive machine, then $y \overset{\$}{\leftarrow} \mathcal{A}(x)$ means we assign y to the output of $\mathcal{A}$ when run with fresh random coins on input x.

If $\mathcal{A}$ and $\mathcal{B}$ are two interactive machines, then let $\mathcal{A} \longleftrightarrow \mathcal{B}$ denote the interaction between these two machines. We will add a text over the arrow $\overset{\text{Setting}}{\longleftrightarrow}$ to denote the case where there are more than 2 machines interacting, but where we wish to focus specifically the two machines $\mathcal{A}$ and $\mathcal{B}$. Let $\mathrm{Out}_\mathcal{A}(\mathcal{A} \leftrightarrow \mathcal{B})$ denote $\mathcal{A}$'s final output in the interaction.

We say that a function $\nu : \mathbb{N} \mapsto [0,1]$ is negligible (denoted $\nu = \mathit{negl.}$) if its inverse is increasing faster than any polynomial. Formally: $\forall c \in \mathbb{N} : \exists \kappa_c \in \mathbb{N} : \forall \kappa \in \mathbb{N} : \kappa > \kappa_c \implies \nu(\kappa) < \kappa^{-c}$.

In this paper, when we talk about negligible functions, we always mean negligible with respect to the security parameter κ. Unless noted otherwise all algorithms and interactive machines are PPT and we assume they implicitly take $\mathbf{1}^\kappa$ as an extra input.

2.2 Record Policies

We inherit the framework of Nishide et al. [NYO08] for access control.

Before issuing the system public key, the issuer chooses $n \in \mathbb{N}$ and $\big(n_i \in \mathbb{N}\big)_{i=1}^n$, the total number of regular categories, and the number of attributes in each category. The issuer may assign labels to these categories and attributes (as we did for Figure 1).

The keys issued to users are associated with a list of attributes $L = \big(L_1, \ldots, L_n\big)$, one from each category: $\forall i \in \mathbb{N}_{n+1}^* : L_i \in \mathbb{N}_{n_i}$.

A database associates a ciphertext policy W to each record it encrypts. W could be thought of as a subset of attributes, however for notational convenience we will segregate the attributes of different categories: we thus define W to be a list of subsets of attributes. We write $W = (W_1, \ldots, W_n)$ where $\forall i \in \mathbb{N}^*_{n+1} : W_i \subset \mathbb{N}_{n_i}$ and where $\varrho \neq 0$ is the database index.

A key is authorized to decrypt a given ciphertext (we also say that the key satisfies the policy of the record/ciphertext) if and only if all attributes in the key are also in the ciphertext policy $\forall i \in \mathbb{N}^*_{n+1} : L_i \in W_i$.

For instance, in Figure 1 the issuer has set $n = 5$ and $\big(n_i\big)_{i=1}^{n} = \big(6, 2, 6, 5, 3\big)$. The ciphertext policy that is shown is $\big(W_i\big)_{i=1}^{n} = \big(\{0, 1\}, \{1\}, \{0, 1, 2\}, \{0, 2\}, \{1, 2\}\big)$. For example Alice, age 23, and full-time predoc in life sciences, is issued a key with the attributes $\big(L_i\big)_{i=1}^{n} = \big(1, 1, 2, 2, 2\big)$. Alice's key satisfies the policy W, she would therefore be able to decrypt the message after interacting with the database. Eve, age 26, who is in the same program as Alice, will receive a key where $L_1 = 2$. Her key does not satisfy the policy, and she would not be able to decrypt the ciphertext (she would notice that only after interacting with the database).

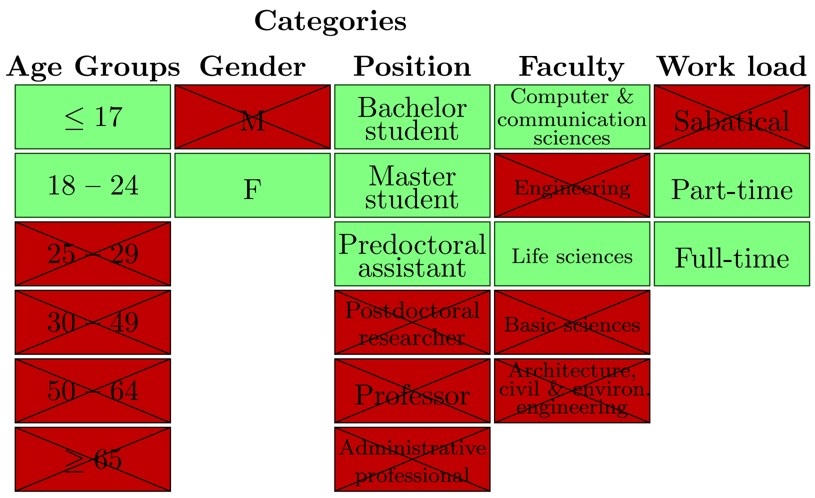

Figure 1: An example ciphertext policy, with labels attached to the categories and attributes. Dark and crossed out boxes represent attributes which are not part of the policy, while light boxes represent attributes which are part of the policy. This policy can be used for example for pictures of a student's party, which should only be accessible by female students aged below 24 in computer, communication or life sciences.

2.3 Algorithms and Protocols

An HABE-OT scheme is a set of eight probabilistic polynomial-time algorithms or protocols:

- IssuerSetup : $\mathcal{I}(n, \{n_i\}_{i=1}^{n}) \xrightarrow{\$} (PI, SI)$.
  The issuer runs this algorithm to generate the system-wide public key PI and corresponding secret key SI. The input to this algorithm is the number of categories n and for each category, the number of attributes n_i in it.

- CheckIssuerKey : $\mathrm{Out}_{\mathcal{U}_\varphi}\big(\mathcal{U}_\varphi(PI) \longleftrightarrow \mathcal{I}(PI, SI)\big) \xrightarrow{\$} b$ and $\mathrm{Out}_{\mathcal{D}_\varrho}\big(\mathcal{D}_\varrho(PI) \leftrightarrow \mathcal{I}(PI, SI)\big) \xrightarrow{\$} b$.
  Upon receiving the public key of the issuer, each user and each database runs this protocol with the issuer, so that the latter can prove in zero-knowledge that he knows the secret key corresponding to his public key. Common input is the issuer's public key. The issuer private input is his secret key. The output is a bit b that says whether the user/database accepts the public key of the issuer or not.

- DatabaseSetup : $\mathcal{D}_\varrho(PI) \xrightarrow{\$} (PD^{(\varrho)}, SD^{(\varrho)})$.
  The database ϱ runs this algorithm to generate its public-key $PD^{(\varrho)}$ and corresponding private key $SD^{(\varrho)}$.

- CheckDatabaseKey : $\mathrm{Out}_{\mathcal{U}_\varphi}\big(\mathcal{U}_\varphi(PI, PD^{(\varrho)}) \longleftrightarrow \mathcal{D}_\varrho(PI, PD^{(\varrho)}, SD^{(\varrho)})\big) \xrightarrow{\$} b$.
  Upon receiving the public key of the database ϱ, each user runs this protocol with the database ϱ, so that the latter can prove in zero-knowledge that it knows the secret key corresponding to its public key. Common input is the issuer's and database's public keys. The database's private input is its secret key. The output is a bit b that says whether the user accepts the public key of the database or not.

- IssueRecord : $\mathcal{D}_\varrho(PI, PD^{(\varrho)}, SD^{(\varrho)}, M^{(\varrho,\psi)}, W^{(\varrho,\psi)}) \xrightarrow{\$} CT^{(\varrho,\psi)}$.
  The database ϱ runs this algorithm to publish a new record ψ. The input are the database's key pair, the issuer's public key, the plaintext $M^{(\varrho,\psi)} \in \mathbb{G}_\mathrm{T}$ of the record, and the policy $W^{(\varrho,\psi)}, \forall i \in \mathbb{N}^*_{n+1} : W^{(\varrho,\psi)}_i \subset \mathbb{N}_{n_i}$. The output is the ciphertext $CT^{(\varrho,\psi)}$.

- CheckRecord : $\mathcal{U}_\varphi(PI, PD^{(\varrho)}, CT^{(\varrho,\psi)}) \xrightarrow{\$} b$.
  Upon receiving a ciphertext $CT^{(\varrho,\psi)}$ from a database ϱ, each user does a check to test whether it is correctly formed or not. The output is a bit b indicating the result of that check.

- IssueKey : $\mathrm{Out}_{\mathcal{U}_\varphi}\big(\mathcal{U}_\varphi(L^{(\varphi)}, PI) \longleftrightarrow \mathcal{I}(\varphi, L^{(\varphi)}, PI, SI)\big) \xrightarrow{\$} SU^{(\varphi)}$.
  The user φ and the issuer run this interactive protocol to generate a new secret key $SU^{(\varphi)}$ for the user φ. Common input are the user index φ (this is an authenticated channel), attributes $L^{(\varphi)}, \forall i \in \mathbb{N}^*_{n+1} : L^{(\varphi)}_i \in \mathbb{N}_{n_i}$ to associate with the key, and the public key of the issuer. The issuer additionally has to provide his secret key as private input. Only the user receives output from this protocol: namely his secret key $SU^{(\varphi)}$.

- Query : $\mathrm{Out}_{\mathcal{U}_\varphi}\big(\mathcal{U}_\varphi(PI, PD^{(\varrho)}, CT^{(\varrho,\psi)}, SU^{(\varphi)}) \longleftrightarrow \mathcal{D}_\varrho(PI, PD^{(\varrho)}, SD^{(\varrho)})\big) \xrightarrow{\$} M'$.
  The user φ queries the database ϱ in order to attempt to decrypt the ciphertext of record ψ. The common input is the public keys of the issuer and database. The user's private input is his secret key and the ciphertext of record ψ. The database's secret input is its private key. Only the user receives output from this protocol: namely the recovered plaintext M'. If decryption was successful, $M' = M^{(\varrho,\psi)}$, and if it failed then $M' = \perp$. The database does not know with which user it interacted with, nor which ciphertext it helped decrypt. The user does not recover the policy $W^{(\varrho,\psi)}$ of the record.

We assume the CRS and group setup have already been generated and are implicitly added as common input to all eight algorithms/protocols.

2.4 Security

We define the security of the HABE-OT protocol through an indistinguishability argument between a real world and an ideal world construct, as introduced by the UC framework [Can00] and reactive systems security [PW00, PW01], and used successfully in several papers [CNS07, CDN09, CDNZ11].

In the real world there are a number of players who run some cryptographic protocols with each other. Some of these players are honest and follow the original protocol. The other, dishonest, players may behave arbitrarily and are controlled by an adversary $\mathcal{A}$. We define an environment $\mathcal{E}$ (a PPT interactive machine) which provides all the inputs and outputs of all the honest players, and interacts arbitrarily with the adversary.

In the ideal world, the players do not run any cryptographic protocols, but interact with a trusted third party T. The latter applies the functionality that the cryptographic protocols are supposed to realize. The honest players do nothing more than relaying messages from the environment to the trusted party, and vice-versa. The environment $\mathcal{E}$ again provides the inputs and outputs for all honest players and interacts arbitrarily with the adversary $\mathcal{S}$ controlling the dishonest players.

We say that a scheme securely implements a scheme if, for every real-world adversary $\mathcal{A}$ (a PPT interactive machine), and every environment $\mathcal{E}$, there exists an ideal-world adversary $\mathcal{S}$ (also a PPT interactive machine) controlling the same parties as $\mathcal{A}$, such that $\mathcal{E}$ cannot distinguish if it is interacting with $\mathcal{A}$ in the real world or $\mathcal{S}$ in the ideal world. More formally, we define the HABE-OT real-world / ideal-world distinguishing game for a PPT algorithm $\mathcal{E}$:

- Before the start of the protocol, the environment $\mathcal{E}$ must select a subset of parties to corrupt, and provide an arbitrary probabilistic polynomial-time (PPT) algorithm describing their behaviour (which we collectively call $\mathcal{A}$).
- After that a challenger flips a random coin b. If $b = 0$ then $\mathcal{E}$ will interact with the honest parties and the corrupted parties he defined in the real world. If $b = 1$, then $\mathcal{E}$ will interact with the honest parties and a simulator $\mathcal{S}$ in the ideal world.
- Thereafter, $\mathcal{E}$ may interact with the honest parties as described later in Section 2.4, and arbitrarily with the corrupted parties. Honest parties will strictly follow their respective protocols; all corrupted will behave as per the PPT algorithm $\mathcal{A}$.
- Finally $\mathcal{E}$ outputs a guess b' of b.

The advantage of $\mathcal{E}$ is defined as being:

$$\mathbf{Adv}^{\text{HABE-OT}}_{\mathcal{E},\mathcal{A}} \overset{\text{def}}{=} \left|\Pr\left[\text{Out}_{\mathcal{E}}(\mathcal{E} \overset{\text{Real}}{\longleftrightarrow} \mathcal{A}) \overset{\$?}{=} 1\right] - \Pr\left[\text{Out}_{\mathcal{E}}(\mathcal{E} \overset{\text{Ideal}}{\longleftrightarrow} \mathcal{S}) \overset{\$?}{=} 1\right]\right|$$

The scheme is secure if

$$\forall \mathcal{A} : \exists \mathcal{S} : \forall \mathcal{E} : \mathbf{Adv}^{\text{HABE-OT}}_{\mathcal{E},\mathcal{A}} = \textit{negl.}$$

We must stress that $\mathcal{E}$ and $\mathcal{A}$ are not the same entity. One way $\mathcal{E}$ can get an advantage in distinguishing between the two worlds is when $\mathcal{A}$ can recover information (such as the plaintext or the policy of a record) it would not be able to get in the ideal world.

We denote messages in the following way: $\langle \mathcal{S}, \mathcal{R}, \textit{str}, \textit{arg1}, \textit{arg2}, \ldots \rangle$ where $\mathcal{S}$ is the sender of the message, $\mathcal{R}$ is the recipient of the message, *str* is a string identifying the type of message and *arg1*, *arg2*, ... are the payload. When the message is broadcasted to all parties in the set $\mathcal{B}$ we write $\langle \mathcal{S}, \forall \mathcal{R} \in \mathcal{B}, \textit{str}, \ldots \rangle$. When communicating over anonymous channels, we put the sender in parenthesis: $\langle (\mathcal{S}), \mathcal{R}, \textit{str}, \ldots \rangle$.

We make the assumption that a user $\mathcal{U}_\varphi$ never receives more than 1 key. If a person gets several keys, we model him as several distinct users.

2.4.1 Real World

We begin by describing the interface between the environment $\mathcal{E}$ and the real-world algorithms/protocols presented in 2.3, when all parties are honest. Unless noted otherwise, we assume all channels are authenticated, secure and integer. All parties are PPT interactive machines. All parties controlled by the adversary $\mathcal{A}$ may deviate arbitrarily from the specifications.

(1) $\mathcal{E}$ decides on the number of categories n and attributes $\left(n_i\right)_{i=1}^{n}$ and sends $\langle \mathcal{E}, \mathcal{I}$, "InitIssuer", $n, \left(n_i\right)_{i=1}^{n} \rangle$. $\mathcal{I}$ sets $n_0 \leftarrow 1$, runs $\mathsf{IssuerSetup}$, and broadcasts PI to every other player: $\langle \mathcal{I}, \forall \mathcal{R} \in (\mathcal{U} \cup \mathcal{D})$, "IssuerKey", $PI \rangle$. Each recipient $\mathcal{R} \in (\mathcal{U} \cup \mathcal{D})$ then runs $\mathsf{CheckIssuerKey}$ with $\mathcal{I}$ (yielding a result b), and sends $\langle \mathcal{R}, \mathcal{E}$, "ReceivedIssuerKey", $n, \left(n_i\right)_{i=0}^{n}, b \rangle$ back to the environment.

(2) Next, $\mathcal{E}$ sends $\langle \mathcal{E}, \forall \mathcal{D}_\varrho \in \mathcal{D}$, "InitDB"$\rangle$. Each $\mathcal{D}_\varrho$ then runs $\mathsf{DatabaseSetup}$, and broadcasts $PD^{(\varrho)}$ to all users: $\langle \mathcal{D}_\varrho, \forall \mathcal{U}_\varphi \in \mathcal{U}$, "DBKey", $PD^{(\varrho)} \rangle$. Each user $\mathcal{U}_\varphi$ then runs $\mathsf{CheckDatabaseKey}$ with $\mathcal{D}_\varrho$ (yielding a result b), and sends $\langle \mathcal{U}_\varphi, \mathcal{E}$, "ReceivedDBKey", $\varrho, b \rangle$ back to the environment.

The steps (3) to (5) may now run in any order, a polynomial number of times.

(3) When $\mathcal{E}$ sends $\langle \mathcal{E}, \mathcal{D}_\varrho$, "IssueRecord", $M, W \rangle$ to some database $\mathcal{D}_\varrho$ (where $M \in \mathbb{G}_\mathrm{T}$ and W is a ciphertext policy); $\mathcal{D}_\varrho$ runs $\mathsf{IssueRecord}$ and broadcasts the resulting ciphertext (with index ψ) to all users and the issuer: $\langle \mathcal{D}_\varrho, \forall \mathcal{R} \in (\mathcal{U} \cup \mathcal{I})$, "NewCiphertext", $CT^{(\varrho,\psi)} \rangle$. Each recipient $\mathcal{R}$ then runs $\mathsf{CheckRecord}$ (yielding a result b) and sends $\langle \mathcal{R}, \mathcal{E}$, "ReceivedCiphertext", $\varrho, \psi, b \rangle$ back to the environment. The issuer $\mathcal{I}$ recovers M and W from the ciphertext $CT^{(\varrho,\psi)}$ using the escrow formulas (see Section 5.1) and sends $\langle \mathcal{I}, \mathcal{E}$, "Escrow", $\varrho, \psi, M, W \rangle$ to the environment.

(4) When $\mathcal{E}$ sends $\langle \mathcal{E}, \mathcal{U}_\varphi$, "IssueKey", $L \rangle$ to some user $\mathcal{U}_\varphi$ (L is a list of attributes, one per category); $\mathcal{U}_\varphi$ sends $\langle \mathcal{U}_\varphi, \mathcal{I}$, "RequestKey", $L \rangle$ to the issuer. $\mathcal{I}$ sends $\langle \mathcal{I}, \mathcal{E}$, "KeyOK?", $\varphi, L \rangle$ to the environment, and receives the reply $\langle \mathcal{E}, \mathcal{I}$, "KeyOK", $b \rangle$. If $b = 0$ then $\mathcal{I}$ aborts the key issuing protocol. Else both $\mathcal{U}_\varphi$ and $\mathcal{I}$ engage in the $\mathsf{IssueKey}$ protocol. Let $b = 0$ if the issuer aborted the protocol and $b = 1$ if it completed successfully. Finally, the user sends $\langle \mathcal{U}_\varphi, \mathcal{E}$, "IssueKeyDone", $b \rangle$ back to the environment.

(5) When $\mathcal{E}$ sends $\langle \mathcal{E}, \mathcal{U}_\varphi$, "Query", $\varrho, \psi \rangle$ to some user $\mathcal{U}_\varphi$; $\mathcal{U}_\varphi$ sends $\langle (\mathcal{U}_\varphi), \mathcal{D}_\varrho$, "Query?"$\rangle$ over an anonymous channel to $\mathcal{D}_\varrho$.[7] $\mathcal{D}_\varrho$ sends $\langle \mathcal{D}_\varrho, \mathcal{E}$, "ProcessQuery?"$\rangle$ to the environment who replies with $\langle \mathcal{E}, \mathcal{D}_\varrho$, "ProcessQuery", $c \rangle$. If $c = 0$, then $\mathcal{D}_\varrho$ aborts the query protocol. Else both U_φ and $\mathcal{D}_\varrho$ engage in the Query protocol over the anonymous channel. If the decryption is successful,[8] the user sends $\langle \mathcal{U}_\varphi, \mathcal{E}$, "QueryResult", $M' \rangle$ back to the environment. If the decryption was not successful, then $\langle \mathcal{U}_\varphi, \mathcal{E}$, "QueryResult", $\perp \rangle$ is sent instead. If the query protocol aborted then $\langle \mathcal{U}_\varphi, \mathcal{E}$, "QueryResult", $\perp\!\!\!\perp \rangle$ is sent.

2.4.2 Ideal World

The interface between the environment $\mathcal{E}$ and the ideal world is necessarily identical to the real-world interface. In the ideal world, no cryptographic protocols are run. An incorruptible trusted third party T performs the functionality that was realized by cryptographic protocols in the real world. All parties may communicate only with T and $\mathcal{E}$. All honest parties are dummies, who do nothing but relay messages between $\mathcal{E}$ and the trusted third party T. All channels are authenticated, secure and integer. The ideal-world adversary S controls all corrupted parties and may interact arbitrarily with $\mathcal{E}$.

[7] One can use onion routing, such as the TOR network, to achieve anonymous communication.

[8] One way to detect if decryption is successful is to use the message M to derive a symmetric key and use an authenticated encryption mode, such as AES-256-GCM, to encrypt the records. If a malicious database outputs garbage instead of a well-formed (symmetric) ciphertext, then with overwhelming probability no honest user will be able to decrypt the ciphertext, which is functionally equivalent to using the most restrictive policy $W_1 = \emptyset$.

(1) $\mathcal{E}$ decides on the number of categories n and attributes $\big(n_i\big)_{i=1}^{n}$ and sends $\langle \mathcal{E}, \mathcal{I}$, "InitIssuer", $n, \big(n_i\big)_{i=1}^{n}\rangle$. $\mathcal{I}$ relays the message $\langle \mathcal{I}, \mathsf{T}$, "InitIssuer", $n, \big(n_i\big)_{i=1}^{n}\rangle$ to T. The latter sets $n_0 \leftarrow 1$ and broadcasts $\langle \mathsf{T}, \forall \mathcal{R} \in (\mathcal{U} \cup \mathcal{D})$, "ReceivedIssuerKey", $n, \big(n_i\big)_{i=0}^{n}, 1\rangle$ to all other players. Each recipient then relays the message back to the environment: $\langle \mathcal{R}, \mathcal{E}$, "ReceivedIssuerKey", $n, \big(n_i\big)_{i=0}^{n}, 1\rangle$.

(2) Next, $\mathcal{E}$ sends $\langle \mathcal{E}, \forall \mathcal{D}_\varrho \in \mathcal{D}$, "InitDB"$\rangle$. Each $\mathcal{D}_\varrho$ then relays the message $\langle \mathcal{D}_\varrho, \mathsf{T}$, "InitDB"$\rangle$ to T. The latter broadcasts $\langle \mathsf{T}, \forall \mathcal{U}_\varphi \in \mathcal{U}$, "ReceivedDBKey", $\varrho, 1, \rangle$ to all users. Each user $\mathcal{U}_\varphi$ then relays the message back to the environment: $\langle \mathcal{U}_\varphi, \mathcal{E}$, "ReceivedDBKey", $\varrho, 1\rangle$.

The steps (3) to (5) may now run in any order, a polynomial number of times.

(3) When $\mathcal{E}$ sends $\langle \mathcal{E}, \mathcal{D}_\varrho$, "IssueRecord", $M, W\rangle$ to some database $\mathcal{D}_\varrho$ (where $M \in \mathbb{G}_\mathrm{T}$ and W is a ciphertext policy); $\mathcal{D}_\varrho$ relays $\langle \mathcal{D}_\varrho, \mathsf{T}$, "IssueRecord", $M, W\rangle$ to T. The latter sets the record index ψ, saves M and W as $M^{(\varrho,\psi)}$ and $W^{(\varrho,\psi)}$, and broadcasts $\langle \mathsf{T}, \forall \mathcal{R} \in (\mathcal{U} \cup \mathcal{I})$, "ReceivedCiphertext", $\varrho, \psi, 1\rangle$ to all users and the issuer. Each recipient $\mathcal{R}$ then sends $\langle \mathcal{R}, \mathcal{E}$, "ReceivedCiphertext", $\varrho, \psi, 1\rangle$ back to the environment. T also sends $\langle \mathsf{T}, \mathcal{I}$, "Escrow", $\varrho, \psi, M^{(\varrho,\psi)}, W^{(\varrho,\psi)}\rangle$ to the issuer. The issuer relays $\langle \mathcal{I}, \mathcal{E}$, "Escrow", ϱ, ψ, $M^{(\varrho,\psi)}, W^{(\varrho,\psi)}\rangle$ back to the environment.

(4) When $\mathcal{E}$ sends $\langle \mathcal{E}, \mathcal{U}_\varphi$, "IssueKey", $L\rangle$ to some user $\mathcal{U}_\varphi$ (L is a list of attributes, one per category); $\mathcal{U}_\varphi$ relays $\langle \mathcal{U}_\varphi, \mathsf{T}$, "IssueKey", $L\rangle$ to T. The latter sends $\langle \mathsf{T}, \mathcal{I}$, "KeyOK?", $\varphi, L\rangle$ to the issuer. The issuer relays $\langle \mathcal{I}, \mathcal{E}$, "KeyOK?", $\varphi, L\rangle$ to the environment, and receives the reply $\langle \mathcal{E}, \mathcal{I}$, "KeyOK", $b\rangle$. The issuer relays $\langle \mathcal{I}, \mathsf{T}$, "KeyOK", $b\rangle$ to T. If $b = 1$, then T saves L as $L^{(\varphi)}$ and sends $\langle \mathsf{T}, \mathcal{U}_\varphi$, "IssueKeyDone", $b\rangle$ back to the user. Finally, the user relays $\langle \mathcal{U}_\varphi, \mathcal{E}$, "IssueKey Done", $b\rangle$ to the environment.

(5) When $\mathcal{E}$ sends $\langle \mathcal{E}, \mathcal{U}_\varphi$, "Query", $\varrho, \psi\rangle$ to some user U_φ; the latter relays $\langle \mathcal{U}_\varphi, \mathsf{T}$, "Query", $\varrho, \psi\rangle$ to T. If $L^{(\varphi)}$ has not been initialized, then T aborts the protocol. T sends $\langle \mathsf{T}, \mathcal{D}_\varrho$, "ProcessQuery?"$\rangle$ to database $\mathcal{D}_\varrho$, the latter relays $\langle \mathcal{D}_\varrho, \mathcal{E}$, "ProcessQuery?"$\rangle$ to the environment who replies with $\langle \mathcal{E}, \mathcal{D}_\varrho$, "ProcessQuery", $c\rangle$. $\mathcal{D}_\varrho$ relays the message $\langle \mathcal{D}_\varrho, \mathsf{T}$, "ProcessQuery", $c\rangle$ to the trusted party. If $c = 0$, then T sends $\langle \mathsf{T}, \mathcal{U}_\varphi$, "QueryResult", $\perp\!\!\!\perp\rangle$ back to $\mathcal{U}_\varphi$. Else if $c = 1$ and $L^{(\varphi)}$ satisfies the policy $W^{(\varrho,\psi)}$, then T sends $\langle \mathsf{T}, \mathcal{U}_\varphi$, "QueryResult", $M^{(\varrho,\psi)}\rangle$. Else if $c = 1$ and $L^{(\varphi)}$ does not satisfy the policy, T sends $\langle \mathsf{T}, \mathcal{U}_\varphi$, "QueryResult", $\perp\rangle$. Finally the user sends respectively $\langle \mathcal{U}_\varphi, \mathcal{E}$, "QueryResult", $\perp\!\!\!\perp\rangle$, $\langle \mathcal{U}_\varphi, \mathcal{E}$, "QueryResult", $M^{(\varrho,\psi)}\rangle$, or $\langle \mathcal{U}_\varphi, \mathcal{E}$, "QueryResult", $\perp\rangle$ back to the environment.

2.5 Security Properties

In this subsection we will give an informal description of the security properties of our scheme.

Database security Users need to interact with (query) the database in order to try to decrypt a ciphertext. Users cannot determine if their key satisfies the access policy of the record before the interaction. After a successful interaction, users cannot deduce anything about the policy except whether their key satisfies it or not.

Cheating and colluding users do not have more power than each user taken individually, in particular they cannot "combine" or "rearrange" the attributes in their keys, and they cannot use the transcript of their interaction with the database to try and decrypt different ciphertexts or with different keys. They cannot deduce the plaintext of a ciphertext if they haven't queried for it with a valid key satisfying the policy. They cannot deduce anything about the policy of a record except of course that each time they query the record's ciphertext with a valid key, they know if that key satisfies the policy of the record.

User security The only information that the database can gather is the number of attempted decryptions. In particular it cannot determine which user queried which ciphertext during any of the queries, what policy the ciphertext was encrypted under, what attributes the user has, and whether the decryption was successful or not. User security is valid even if the database colludes with the issuer and some other users.

If the query protocol completed successfully, honest users are guaranteed that (1) if the decryption was successful, their key satisfies the policy of the ciphertext and (2) if the decryption was unsuccessful, their key does not satisfy the policy of the ciphertext.

2.5.1 Limitations

Issuer escrow The issuer can extract the plaintext and the policy of all the databases' ciphertexts without interacting with them. This functionality is an unfortunate side effect of the HABE scheme.

This functionality makes our scheme slightly weaker than [CDNZ11]: in their scheme, the issuer can generate a credential for himself containing all attributes of the system, but he must interact with the database to decrypt.

If we decide to combine the roles of the issuer and the database in our scheme,[9] then this functionality does not cause any problems anymore.

3 Preliminaries

3.1 Bilinear Maps

A bilinear map is a function $\mathrm{e} : \mathbb{G}_1 \times \mathbb{G}_2 \mapsto \mathbb{G}_\mathrm{T}$, where $\mathbb{G}_1$, $\mathbb{G}_2$ and $\mathbb{G}_\mathrm{T}$ are abelian groups of prime order p. Let $g_1 \in \mathbb{G}_1 \setminus \{1\}$ and $g_2 \in \mathbb{G}_2 \setminus \{1\}$ denote generators of $\mathbb{G}_1$ and $\mathbb{G}_2$. The bilinear map must satisfy the following properties:

- **Bilinearity:** $\forall a, b \in \mathbb{Z}_p : \quad \mathrm{e}(g_1^a, g_2^b) = \mathrm{e}(g_1, g_2)^{ab}$.
- **Non-degeneracy:** $g_\mathrm{T} \overset{\text{def}}{=} \mathrm{e}(g_1, g_2) \neq 1$.

The element g_T is of course a generator of $\mathbb{G}_\mathrm{T}$. Throughout this document, we will use multiplicative notation for the groups $\mathbb{G}_1$, $\mathbb{G}_2$ and $\mathbb{G}_\mathrm{T}$. We assume that all algorithms implicitly take a description of the groups as an extra input parameter.

3.2 Assumptions

Decisional Diffie-Hellman (DDH)

Let $\mathbb{G}$ be either $\mathbb{G}_1$, $\mathbb{G}_2$ or $\mathbb{G}_\mathrm{T}$ (and let g be the corresponding generator of the group, viz. g_1, g_2, g_T). Let $a, b, z \overset{\$}{\leftarrow} \mathbb{Z}_p$. DDH is hard in $\mathbb{G}$ if for every PPT algorithm $\mathcal{A}$:

$$\mathbf{Adv}_{\mathbb{G}}^{\mathrm{DDH}} \overset{\text{def}}{=} \left| \Pr\left[\mathcal{A}(g, g^a, g^b, \underline{g^{ab}}) \overset{\$?}{=} 1\right] - \Pr\left[\mathcal{A}(g, g^a, g^b, \underline{g^z}) \overset{\$?}{=} 1\right] \right| = \mathit{negl.}$$

[9] Thus forfeiting the possibility of having multiple independent databases.

Symmetric External Diffie-Hellman (SXDH) Assumption

We say that the SXDH assumption holds if DDH holds in $\mathbb{G}_1$, $\mathbb{G}_2$ and $\mathbb{G}_\mathrm{T}$. This assumption holds only for type-3 [GPS08] bilinear maps: this means there exists no efficiently computable homomorphism from $\mathbb{G}_1$ to $\mathbb{G}_2$ or vice-versa. It is believed SXDH holds in certain subgroups of MNT elliptic curves.

Decisional Bilinear Diffie-Hellman (DBDH) Assumption

Let $z_1, z_2, z_3, z \overset{\$}{\leftarrow} \mathbb{Z}_p$, $g_1 \in \mathbb{G}_1^*$, $g_2 \in \mathbb{G}_2^*$. The DBDH assumption holds in $\mathbb{G}_1, \mathbb{G}_2$ if for every PPT algorithm $\mathcal{A}$ [NYO08]:

$$\begin{aligned}\mathbf{Adv}^{\mathrm{DBDH}}_{\mathbb{G}_1,\mathbb{G}_2} \overset{\mathrm{def}}{=} \Big|\Pr\Big[\mathcal{A}\Big(g_1, g_1^{z_1}, g_1^{z_2}, g_1^{z_3}, g_2, g_2^{z_1}, g_2^{z_2}, g_2^{z_3}, \underline{g_\mathrm{T}^{z_1 z_2 z_3}}\Big) \overset{\$?}{=} 1\Big] \\ - \Pr\Big[\mathcal{A}\Big(g_1, g_1^{z_1}, g_1^{z_2}, g_1^{z_3}, g_2, g_2^{z_1}, g_2^{z_2}, g_2^{z_3}, \underline{g_\mathrm{T}^{z}}\Big) \overset{\$?}{=} 1\Big]\Big| = \mathit{negl.}\end{aligned}$$

Note that the DBDH assumption was originally made for symmetrical (type 1) pairings only. Using the transformation introduced by Chatterjee et al. [CM09], we have generalized it type-3 pairings.[10]

Decision Linear (D-Lin) Assumption

Let $z_1, z_2, z_3, z_4, z \overset{\$}{\leftarrow} \mathbb{Z}_p$. The D-Lin assumption holds in the input groups if for every PPT algorithm $\mathcal{A}$ [NYO08]:

$$\begin{aligned}\mathbf{Adv}^{\text{D-Lin}}_{\mathbb{G}_1,\mathbb{G}_2} \overset{\mathrm{def}}{=} \Big|\Pr\Big[\mathcal{A}\Big(g_1, g_1^{z_1}, g_1^{z_2}, g_1^{z_2 z_4}, g_1^{z_3+z_4}, g_2, g_2^{z_1}, g_2^{z_2}, \underline{g_1^{z_1 z_3}}\Big) \overset{\$?}{=} 1\Big] \\ - \Pr\Big[\mathcal{A}\Big(g_1, g_1^{z_1}, g_1^{z_2}, g_1^{z_2 z_4}, g_1^{z_3+z_4}, g_2, g_2^{z_1}, g_2^{z_2}, \underline{g_1^{z}}\Big) \overset{\$?}{=} 1\Big]\Big| = \mathit{negl.}\end{aligned}$$

Note that the D-Lin assumption was originally made for symmetrical (type 1) pairings only, but we can generalize it to type-3 pairings in the same way as for DBDH[CM09].[11]

ℓ-Simultaneous Flexible Pairing (SFP) Assumption

Let $A, B \overset{\$}{\leftarrow} \mathbb{G}_1$, $\tilde{A}, \tilde{B} \overset{\$}{\leftarrow} \mathbb{G}_2$ and $g_Z, f_Z, g_R, f_U \overset{\$}{\leftarrow} \mathbb{G}_1^*$. For $j \in \mathbb{N}^*_{\ell+1}$ let $P_j \overset{\mathrm{def}}{=} \big(Z_j, R_j, S_j, T_j, U_j, V_j, W_j\big)$ that satisfies:

$$\mathrm{e}(A, \tilde{A}) = \mathrm{e}(g_Z, Z_j)\,\mathrm{e}(g_R, R_j)\,\mathrm{e}(S_j, T_j) \text{ and} \tag{1}$$

$$\mathrm{e}(B, \tilde{B}) = \mathrm{e}(f_Z, Z_j)\,\mathrm{e}(f_U, U_j)\,\mathrm{e}(V_j, W_j). \tag{2}$$

We say that the ℓ-SFP assumptions holds in $\mathbb{G}_1$ if for all PPT algorithms $\mathcal{A}$ [AFG+10, AHO10]:

$$\mathbf{Adv}^{\ell-\mathrm{SFP}}_{\mathbb{G}_1} \overset{\mathrm{def}}{=} \Pr\Big[\mathcal{A}\Big(g_Z, f_Z, g_R, f_U, A, B, \tilde{A}, \tilde{B}, \big(P_j\big)_{j=1}^{\ell}\Big) \overset{\$?}{=} P_{\ell+1}\Big] = \mathit{negl.}$$

[10] The original DBDH problem for symmetrical pairings $\mathbb{G}_1 = \mathbb{G}_2$ asks, given $g_1, g_1^{z_1}, g_1^{z_2}, g_1^{z_3}$ to distinguish $g_\mathrm{T}^{z_1 z_2 z_3}$ from random. The transformation simply adds $g_2, g_2^{z_1}, g_2^{z_2}, g_2^{z_3}$ to the input in the asymmetrical setting, following the observation that we would have been able to compute these group elements anyway in the symmetrical setting.

[11] The original formulation of D-Lin for symmetrical settings is: given $g_1, g_1^{z_1}, g_1^{z_2}, g_1^{z_1 z_3}, g_1^{z_2 z_4}$ distinguish $g_1^{z_3+z_4}$ from random. The following formulation is equivalent: given $g_1, g_1^{z_1}, g_1^{z_2}, g_1^{z_2 z_4}, g_1^{z_3+z_4}$ distinguish $g_1^{z_1 z_3}$ from random. To transform this to the asymmetric setting, we add $g_2, g_2^{z_1}, g_2^{z_2}$ to the input (which we would have been able to compute anyway in the symmetric setting).

where $P_{\ell+1}$ satisfies both Equations 1 and 2 and where $Z_{\ell+1} \neq 1$ and $\forall i \in \mathbb{N}^*_{\ell+1} : Z_{\ell+1} \neq Z_i$.

ℓ-SFP in $\mathbb{G}_2$ is defined analogously by exchanging the groups $\mathbb{G}_1$ and $\mathbb{G}_2$ in the above definition.

We also note that this assumption is parameterized by ℓ (unlike the other assumptions which are static), where a larger ℓ means a stronger assumption. This assumption was proven to hold in the generic bilinear group model [AHO10], as long as $\ell \ll \sqrt{p}$ (quadratic bound).

Generic Bilinear Group Model

The generic bilinear group model is an idealization of the groups and associated bilinear map used in a given construction. Note that the generic group model subsumes all of the assumptions made in the previous paragraphs.

The plain generic group model was introduced by [Sho97] and extended to bilinear groups in [BB08].

In the generic group model all elements of $\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T$ are encoded as arbitrary unique strings, and it is assumed that no operation other than equality testing can be performed on this representation. An example would be to encode the group elements sequentially, in the order the adversary sees them. The adversary has access to several oracles, which he needs to query to perform his computations: three oracles to do multiplication in the three groups respectively, and one oracle to compute the bilinear pairing.

Roughly speaking, if a scheme is proven secure in the generic bilinear group model, then any successful attack against the scheme must exploit nontrivial structure of the groups used (or a flaw in the proof). However, like the random oracle model, the generic group model has attracted a lot of criticism [KM06].

3.3 Perfect Zero-knowledge Proofs of Knowledge

A zero-knowledge proof of knowledge (ZKPoK) [CDM00] is a protocol in which a prover $\mathcal{P}$ can convince a verifier $\mathcal{V}$ that he knows a secret—for instance, the discrete logarithm of a certain element, the discrete-logarithm–based representation of a certain element to certain bases, a pre-image of a bilinear pairing—without disclosing these to the verifier.

We will use the notation introduced by Camenisch and Stadler[CS97b] when we need to perform a ZKPoK in a protocol. For example:

$$\mathsf{ZKPoK}\{(\boxed{\alpha}, \boxed{\beta}) : y = g^{\boxed{\alpha}} \wedge z = g^{\boxed{\beta}} h^{\boxed{\alpha}}\}$$

is used for proving the knowledge of the discrete logarithm of y to the base g, and a representation of z to the bases g and h such that the h-part of this representation is equal to the discrete logarithm of y to the base g [CS97b, p. 414]. We will use the convention that letters in the first parenthesis (which are also boxed in red[12] throughout the equation) denote the elements whose knowledge is proven, and all other letters denote elements which are known to the verifier.

With ZKPoK it is possible to work with statements of the form:

$$\mathsf{ZKPoK}\Big\{\Big((\boxed{x_i} \in \mathbb{Z}_p)_{i=0}^{u_x}, (\boxed{\gamma_i} \in \mathbb{G}_1)_{i=0}^{u_g}, (\boxed{\eta_i} \in \mathbb{G}_2)_{i=0}^{u_h} \Big) : \bigwedge_{i=0}^{n_g} \Big(G_i = \prod_{j=0}^{n_{1,i}} g_{1,i,j}^{\boxed{x_{\nu_{1,i,j}}}} \Big) \tag{3}$$

$$\bigwedge_{i=0}^{n_h} \Big(H_i = \prod_{j=0}^{n_{2,i}} h_{2,i,j}^{\boxed{x_{\nu_{2,i,j}}}} \Big) \bigwedge_{i=0}^{n_t} \Big(T_i = \prod_{j=0}^{n_{3,i}} t_{3,i,j}^{\boxed{x_{\nu_{3,i,j}}}} \prod_{j=0}^{n_{4,i}} \mathrm{e}(g_{4,i,j}, \boxed{\eta_{\nu_{4,i,j}}}) \prod_{j=0}^{n_{5,i}} \mathrm{e}(\boxed{\gamma_{\nu_{5,i,j}}}, h_{5,i,j}) \Big) \Big\},$$

[12] Unless of course you are currently reading this document from a black-and-white printout :-)

where $t_{3,i,j}, T_i \in \mathbb{G}_{\mathrm{T}}$, $g_{1,i,j}, g_{4,i,j}, G_i \in \mathbb{G}_1$, $h_{2,i,j}, h_{5,i,j}, H_i \in \mathbb{G}_2$ and $\nu_{1,i,j}, \nu_{2,i,j}, \nu_{3,i,j} \in \mathbb{N}_{u_x+1}$, $\nu_{4,i,j} \in \mathbb{N}_{u_g+1}$, $\nu_{5,i,j} \in \mathbb{N}_{u_h+1}$ and $u_x, u_g, u_h, n_g, n_h, n_t, \big(n_{k,i}\big)_{k=1}^{5} \in \mathbb{N}$ are known to both parties.

In Appendix C we give more details about the construction and the security properties of ZKPoKs.

3.4 Non-interactive Zero-knowledge Proofs

A non-interactive zero-knowledge (NIZK) proof allows a prover $\mathcal{P}$ to certify that a certain statement is satisfiable, without revealing a satisfiable assignment in the proof (however $\mathcal{P}$ usually needs to know such a satisfiable assignment to be able to compute the proof). The proof he issues can be checked offline by a verifier $\mathcal{V}$.

Common Reference String

In the Common Reference String (CRS) model we assume that a set of values (the CRS) which were generated according to some distribution $\mathfrak{D}$ are made available to all participants at the start of the protocol. It is also assumed that no participants gets to know any "extra information" or trapdoor on that CRS, and that a trusted third party generates the CRS.

When a simulator $\mathcal{S}$ has black-box access to an interactive machine $\mathcal{A}$, he can dictate the CRS that $\mathcal{A}$ will use in the protocol. Usually $\mathcal{S}$ generates the CRS such that he knows the discrete logarithm of all its values (trapdoor) or generate it with a different distribution $\mathfrak{D}'$.

Non-interactive proofs are impossible in the standard model [GO94, BFM88], i.e. without CRS or random oracles.

Groth-Sahai Proofs

In our construction, we will work with a subset of Groth-Sahai (GS) proofs [GS08, GSW10] which allows us to prove the satisfiability of the following classes of statements:

$$\mathsf{NIZK}\Big\{\Big(\big\{x_i \in \mathbb{Z}_p\big\}_{i=0}^{u}\Big) : \bigwedge_{i=0}^{n} \Big(G_i = \prod_{j=0}^{n_i} g_{i,j}^{x_{\nu_{i,j}}}\Big)\Big\},$$

where $G_i, g_{i,j} \in \mathbb{G}_1$ and $\nu_{i,j} \in \mathbb{N}_{u+1}$ and $u, n, n_i \in \mathbb{N}$ are known to both $\mathcal{P}$ and $\mathcal{V}$. GS proofs require the SXDH assumption to hold.

GS proofs are not proofs of knowledge, since it is possible to combine several GS proofs to construct a GS proof of a related statement (as used in [CDNZ11]).

In Appendix D we give more details about the construction and the security properties of GS proofs.

3.5 Hidden–ciphertext-policy Attribute-based Encryption

A hidden–ciphertext-policy attribute-based encryption (HABE) scheme is a set of four algorithms [NYO08]:

- $\mathsf{Setup}(n, \big(n_i\big)_{i=1}^{n}) \xrightarrow{\$} \{PK, MK\}$.
  This algorithm generates the system public key PK and master secret key MK. The inputs are the number of categories and attributes par category (see also Section 2.2).

- $\mathsf{KeyGen}(L^{(\varphi)}, PK, MK) \xrightarrow{\$} SK_L^{(\varphi)}$.
  This algorithm takes a list of attributes $L^{(\varphi)}$ (one per category: $L^{(\varphi)} = \{L_1^{(\varphi)} \in N_{n_1}, L_2^{(\varphi)} \in \mathbb{N}_{n_2}, \dots, L_n^{(\varphi)} \in \mathbb{N}_n\}$) and generates a user secret key $SK_L^{(\varphi)}$ containing these. The master secret key is required for this step.

- $\mathsf{Encrypt}(M, W, PK) \xrightarrow{\$} CT$.
  This algorithm takes a plaintext $M \in \mathbb{G}_\mathrm{T}$ and a ciphertext policy W—where $W = \{W_1 \subset \mathbb{N}_{n_1}, W_2 \subset \mathbb{N}_{n_2}, \ldots, W_n \subset \mathbb{N}_n\}$—and generates a ciphertext CT. Only the system public key PK is required to encrypt.

- $\mathsf{OfflineDecrypt}(CT, SK_L^{(\varphi)}, PK) \to M'$.
  This algorithm takes a key and a ciphertext and returns the result of the decryption M'. If the key satisfies the ciphertext policy (i.e., $\forall i \in \mathbb{N}^*_{n+1} : L_i^{(\varphi)} \in W_i$) then this algorithm outputs the plaintext M. If the key doesn't satisfy the policy, then with overwhelming probability $M' \neq M$.

The category and attribute setup, the attributes in the key, and the structure of the ciphertext policies are exactly as described in Section 2.2. In Appendix A we give more details about the construction of both HABE schemes of [NYO08].

3.5.1 Security Game for HABE

A HABE scheme is (match-concealing) secure if no PPT interactive machine $\mathcal{A}$ can win the following game with probability non-negligibly higher than 1/2 [Nis08]:

(1) The challenger runs Setup and hands over PK to $\mathcal{A}$.

(2) $\mathcal{A}$ submits a list of attributes L (one per category) to the challenger. The challenger gives the adversary the corresponding secret key SK_L. This phase may be repeated polynomially many times.

(3) $\mathcal{A}$ submits two messages $M^{(0)}$, $M^{(1)}$ and two ciphertext policies $W^{(0)}$, $W^{(1)}$ to the challenger. The latter flips a coin b, encrypts $M^{(b)}$ under policy $W^{(b)}$ and sends the resulting ciphertext to $\mathcal{A}$.

(4) Phase 2 is repeated.

(5) The adversary outputs a guess b' of b. The adversary wins if $b' = b$ and if there was no trivial way for him to win this game: If $\mathcal{A}$ obtained one key SK_L which satisfies either $W^{(0)}$ or $W^{(1)}$, then it must be the case that:

- $M^{(0)} = M^{(1)}$ and
- each key satisfies either: both policies $W^{(0)}$ and $W^{(1)}$, or none of the two policies.

We define the advantage of $\mathcal{A}$ to be: $\mathbf{Adv}_{\mathcal{A}}^{\mathrm{HABE}} \overset{\mathrm{def}}{=} \max(\Pr[\mathcal{A} \text{ wins HABE game}] - 1/2, 0)$.

Our construction makes extensive use of the second HABE scheme of Nishide et al. [NYO08, Nis08], which is secure[13] in the generic bilinear group model.

[13]See [Nis08, p. 23], since [NYO08] only says that their scheme is *selectively* secure.

3.6 Structure-Preserving Signatures

In our construction we use the basic signature scheme of Abe et al. [AFG$^+$10, Section 5.2] for signing a vector of group elements. That scheme is existentially unforgeable against adaptive chosen message attacks (see Section E.1) if the Simultaneous Flexible Pairing assumption (see Section 3.2) holds. Unlike some other signature schemes, like the Boneh-Boyen signatures [BB08] or credentials (BBS+ signature) [ASM06, CL04], which have a cubic bound in the generic group model, this scheme enjoys an optimal quadratic bound.[14] Also this signature scheme does not require the signer to know the discrete logarithm of the group elements he is signing.

In Appendix E we show how to generate a private signing key ($\mathsf{SigKeyGen}$), how to prove in zero-knowledge that you know the signing key corresponding to a verification key vk ($\mathsf{ZKPoK}_{\text{SigKey}}(vk)$), how to sign and verify a message (Sign, $\mathsf{SigVerify}$), and finally how to re-randomize a signature ($\mathsf{SigRerand}$). We note that a signature consists of 7 group elements in $\mathbb{G}_1$ or $\mathbb{G}_2$. The re-randomization algorithm allows you to re-randomize these group elements in such a way that four of these group elements become totally independent from the other three and the message that was signed. This re-randomization enables us to use standard ZKPoKs to prove possession of a signature in zero-knowledge.

We use this signature scheme in the query protocol. It allows the user to blind group elements while at the same time proving in zero-knowledge to the database that he knows the unblinded values. The user's anonymity is thus preserved, and the database has the guarantee that it does not help to decrypt invalid ciphertexts.

4 Our Construction

4.1 Main Idea

We modify the second construction of Nishide et al. [NYO08] as follows:

Zeroth attribute category In addition to the n regular categories, the issuer creates an additional zeroth category, and creates one attribute—which he labels "Issuer"—in that category. Let $A_{0,0} = g_1^{a_{0,0}}$ be the public-key component associated to that attribute ($a_{0,0}$ is the private key component). All the users' keys he issues will contain this "Issuer" attribute (that is $L_0 = 0$).

Each database $\mathcal{D}_\varrho$ that joins the system, extends the zeroth category with a new attribute, which it labels "Database ϱ": it publishes a public-key component $A_{0,\varrho} = A_{0,0}^{k_\varrho}$ for that new attribute, where k_ϱ is part of the database's private key.

When publishing a ciphertext, $\mathcal{D}_\varrho$ publishes the ciphertext component $C_{0,\varrho,2}$ corresponding to its own "Database ϱ" attribute, but not the component $C_{0,0,2}$ corresponding to the "Issuer" attribute (it also doesn't publish the ciphertext components corresponding to the other databases' attributes). Due to the properties of the HABE scheme and the way the new attribute is defined, we have that $C_{0,0,2} = C_{0,\varrho,2}^{k_\varrho^{-1}}$.

Users cannot decrypt offline anymore, since the ciphertext component for the "Issuer" attribute is in none of the ciphertexts issued by any of the databases: he cannot compute the term $\mathrm{e}(C_{0,L_0,2}, D_{0,2})$ in Equation 6. A user who wishes to decrypt a record needs to query the database $\mathcal{D}_\varrho$ that issued that record. The user then blinds the "Issuer" component $D_{0,2}$ of his key with a random value k_d, and the "Database ϱ" component of the ciphertext $C_{0,\varrho,2}$ with a random value k_c, yielding $D' \leftarrow D_{0,2}^{k_d}$ and $C' \leftarrow C_{0,\varrho,2}^{k_c}$, and sends these to the database.

[14]If a signature scheme has a cubic bound in the generic group model, it should be secure as long as less than ℓ messages with $\ell \ll \sqrt[3]{p}$ are signed. With a quadratic bound, we have $\ell \ll \sqrt{p}$ instead.

The latter transforms the blinded “Database ϱ” ciphertext component into a blinded “Issuer” ciphertext component with the help of its private key $C_{0,0,2}^{k_c} = C'^{k_\varrho^{-1}}$, bilinearly pairs the result with the blinded key component of the user $P' \leftarrow \mathrm{e}(C'^{k_\varrho^{-1}}, D')$, and sends the result P' back. The user can unblind the result $\mathrm{e}(C_{0,0,2}, D_{0,2}) \leftarrow P'^{k_c^{-1} k_d^{-1}}$, and with that value, he can proceed with the HABE decryption.

In the rest of the paper, when we say that a key (with attribute list L) *satisfies* the policy W of a record/ciphertext, we mean that each attribute in the key *excluding the “Issuer” attribute* is in the set of attributes of the policy: $\forall i \in \mathbb{N}_{n+1}^* : L_i \in W_i$.

Signing group elements In the query protocol, the database must have the assurance that the blinded values C' and D' sent by the user have been computed honestly, lest it becomes a decryption oracle. For this, the issuer signs the “Issuer” key component $D_{0,2}$ of each user, and each database signs the “Database ϱ” ciphertext component $C_{0,\varrho,2}$ in all ciphertexts it issues with the structure preserving signatures by Abe et al. [AHO10, AFG$^+$10] (introduced in Section 3.6). Users can then prove in zero knowledge that they know the values $C_{0,\varrho,2}$ and $D_{0,2}$ they blinded, and that said values are part of a legitimately issued key, resp. ciphertext. The issuer and each database create a signing/verification key pair together with their public/secret key for that purpose.

Interactive key issuing protocol Additionally we transform the key issuing algorithm in the scheme of Nishide et al. into an interactive protocol, in which the key of a user is computed jointly between the issuer and the user. In this protocol we use the homomorphic property of the El-Gamal cryptosystem so that the issuer and the user jointly compute the random values $g_2^{\lambda_i}$ in the user’s key, in such a way that neither party can recover the discrete logarithm λ_i.

The rationale for this is to avoid that a dishonest issuer and database craft a malicious ciphertext that can be decrypted by a user whose key doesn’t satisfy the policy.[15]

Non-interactive proof in ciphertext When issuing a ciphertext, the database $\mathcal{D}_\varrho$ also publishes a Groth-Sahai non-interactive proof certifying that the ciphertext was computed honestly.[16]

4.2 The Construction

We now present the implementation of all algorithms and protocols described in Section 2.3 in our scheme in detail.

4.2.1 Group Setup

We assume all participants agree on the group parameters and that these have been generated in a trusted manner:

$$[p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_\mathrm{T}, g_1 \in \mathbb{G}_1, g_2 \in \mathbb{G}_2, \mathrm{e}] \leftarrow \mathrm{Gen}(1^\kappa).$$

We denote $g_\mathrm{T} \overset{\mathrm{def}}{=} \mathrm{e}(g_1, g_2)$.

[15]It is crucial that this case never happens if we want our security proof to work.
[16]This is required for our security proof to work.

4.2.2 CRS Setup

A trusted third party generates a *CRS* for the databases' GS proofs:

$$\mathfrak{a}, \mathfrak{t} \xleftarrow{\$} \mathbb{Z}_p^*, \qquad \mathfrak{U}_{1,2} \leftarrow g_2^{\mathfrak{a}}, \qquad \mathfrak{U}_{2,1} \leftarrow g_2^{\mathfrak{t}}, \qquad \mathfrak{U}_{2,2} \leftarrow g_2^{\mathfrak{a}\mathfrak{t}}.$$

The CRS is $\{\mathfrak{U}_{1,2}, \mathfrak{U}_{2,1}, \mathfrak{U}_{2,2}\}$. The parameters $\mathfrak{a}$ and $\mathfrak{t}$ must be kept secret from everybody (CRS model). All algorithms and protocols receive the CRS as an extra (implicit) input.

4.2.3 Issuer Setup

The issuer chooses the number of categories n and the number of different attributes possible for each category $\{n_i\}_{i=1}^n$. He sets $n_0 = 1$ and computes:

$$\Big\{\{a_{i,t} \xleftarrow{\$} \mathbb{Z}_p^*\}_{t=0}^{n_i-1}\Big\}_{i=0}^n, \qquad w, \beta \xleftarrow{\$} \mathbb{Z}_p^*, \quad Y \leftarrow g_{\mathrm{T}}^w, \quad B \leftarrow g_1^\beta,$$
$$\Big\{\{A_{i,t} \leftarrow g_1^{a_{i,t}}\}_{t=0}^{n_i-1}\Big\}_{i=0}^n, \qquad \{sgk_I, vk_I\} \leftarrow \mathsf{SigKeyGen}_{\mathbb{G}_2}().$$

The secret key *SI* of the issuer is $\{w, \beta, \{\{a_{i,t}\}_{t=0}^{n_i-1}\}_{i=0}^n, sgk_I\}$.
His public key *PI* is $\{Y, B, \{\{A_{i,t}\}_{t=0}^{n_i-1}\}_{i=0}^n, vk_I, n, \{n_i\}_{i=0}^n\}$.

4.2.4 Check Issuer Key

Everybody who receives the public key checks that Y, B and all $A_{i,t}$ are not the identity element and are of the correct order p, and then requests that the issuer does the following proof knowledge of his private and verification key:

$$\mathsf{ZKPoK}_1\Big\{\Big(\boxed{w}, \boxed{\beta}, \{\{\boxed{a_{i,t}}\}_{t=0}^{n_i-1}\}_{i=0}^n\Big) : Y = g_{\mathrm{T}}^{\boxed{w}} \wedge B = g_1^{\boxed{\beta}} \bigwedge_{i=0}^{n}\Big(\bigwedge_{t=0}^{n_i-1} A_{i,t} = g_1^{\boxed{a_{i,t}}}\Big)\Big\}$$
$$\wedge\ \mathsf{ZKPoK}_{\mathrm{SigKey}}(vk_I).$$

4.2.5 Database Setup

Let ϱ be the implicit database index. We will add a superscript $^{(\varrho)}$ to a variable to distinguish between the variables of different databases. We may omit that superscript if it is clear from the context whose database's variable we mean.

The database ϱ computes:

$$k_\varrho \xleftarrow{\$} \mathbb{Z}_p \setminus \{a_{0,0}\}, \qquad A_{0,\varrho} \leftarrow A_{0,0}^{k_\varrho} = g_1^{a_{0,0}k_\varrho}, \qquad \{sgk_\varrho vk_\varrho\} \leftarrow \mathsf{SigKeyGen}_{\mathbb{G}_1}().$$

The private key $SD^{(\varrho)}$ of database ϱ is: $\{k_\varrho, sgk_\varrho\}$.
The public key $PD^{(\varrho)}$ is: $\{A_{0,\varrho}, vk_\varrho\}$.
This effectively defines another attribute for the zeroth category: $a_{0,\varrho} = a_{0,0}k_\varrho$.

4.2.6 Check Database Key

All users receive $PD^{(\varrho)}$. They are required to check that $A_{0,\varrho} \neq 1$, that the order of $A_{0,\varrho}$ is p, and request that the database does the following proof of knowledge about his private and verification key:

$$\mathsf{ZKPoK}_2\{(\boxed{k_\varrho}) : A_{0,\varrho} = A_{0,0}^{\boxed{k_\varrho}}\} \wedge \mathsf{ZKPoK}_{\mathrm{SigKey}}(vk_\varrho).$$

4.2.7 Issue Record

Let ψ be the implicit record index. We might add a superscript ${}^{(\psi)}$ to variables in the same way as for databases. Let $M \in \mathbb{G}_T$ be the plaintext to encrypt, and let $W^{(\psi)} = [W_0^{(\psi)}, \dots, W_n^{(\psi)}]$, where $\forall i : W_i \subset \mathbb{N}_{n_i}$, be the hidden ciphertext policy.

The database computes the following values:

$$\left\{r_i^{(\psi)} \xleftarrow{\$} \mathbb{Z}_p^*\right\}_{i=0}^n, \qquad r^{(\psi)} \leftarrow \sum_{i=0}^n r_i, \qquad \hat{C}^{(\psi)} \leftarrow MY^r,$$

$$\left\{\{\epsilon_{i,t} \leftarrow 0\}_{\forall t \in W_i}\right\}_{i=1}^n, \qquad \left\{\{\epsilon_{i,t} \xleftarrow{\$} \mathbb{Z}_p^*\}_{\forall t \in \mathbb{N}_{n_i} \setminus W_i}\right\}_{i=1}^n, \qquad C_0^{(\psi)} \leftarrow B^r,$$

$$\left\{C_{i,1}^{(\psi)} \leftarrow g_1^{r_i}\right\}_{i=0}^n, \qquad \left\{\left\{C_{i,t,2}^{(\psi)} \leftarrow A_{i,t}^{r_i} g_1^{\epsilon_{i,t}}\right\}_{t=0}^{n_i-1}\right\}_{i=1}^n, \qquad C_{0,\varrho,2}^{(\psi)} \leftarrow A_{0,\varrho}^{r_0},$$

$$\sigma^{(\psi)} \leftarrow \mathsf{Sign}(C_{0,\varrho,2}, sgk_{\text{DB}}).$$

The ciphertext $CT^{(\psi)}$ is: $\left\{\sigma, \hat{C}, C_0, C_{0,1}, C_{0,\varrho,2}, \{C_{i,1}, \{C_{i,t,2}\}_{t=0}^{n_i-1}\}_{i=1}^n\right\}$.

The database generates a NIZK GS proof that the record was computed correctly (i.e., such that the issuer can recover the record correctly when using the escrow functionality). We are required to do a non-interactive proof here, as doing an interactive one would increase the asymptotic complexity of the pre-computation time to $O(N_{\text{records}} N_{\text{users}})$:

$$\mathsf{NIZK}_3\left\{(\{\boxed{r_i}\}_{i=0}^n) : \bigwedge_{i=0}^n \left(C_{i,1} = g_1^{\boxed{r_i}}\right) \wedge C_0 = \prod_{i=0}^n B^{\boxed{r_i}} \wedge C_{0,\varrho,2} = A_{0,\varrho}^{\boxed{r_0}}\right\}.$$

At this point we would like to make a few observations:

- We see that for the zeroth category the database only publishes the ciphertext component $C_{0,\varrho,2}$ corresponding to its "Database ϱ" attribute, and not the component $C_{0,0,2}$ for the "Issuer" attribute. No user will thus be able to decrypt the record without help from the database.

- The database can easily recover the "Issuer" ciphertext component with its private key: $C_{0,0,2}^{(\psi)} \leftarrow (C_{0,\varrho,2}^{(\psi)})^{k_\varrho^{-1}}$.

4.2.8 Check Record

When they receive the record, all users are required to check that all group elements are of order p, that $C_{0,\varrho,2}$ is not the neutral element, and that the GS proof and the signature on $C_{0,\varrho,2}$ are correct.

4.2.9 Issue Key

Let φ be the implicit user index. We might add a superscript ${}^{(\varphi)}$ to variables in the same way as for databases and records.

Let $L^{(\varphi)} = [L_0^{(\varphi)}, \dots, L_n^{(\varphi)}]$ where $\forall i : L_i \in \mathbb{N}_{n_i}$ be the attribute list that will be associated to the generated key. We necessarily have $L_0 = 0$, as the issuer defined only a single attribute for the zeroth category. We will assume that the user chooses the value of L and that the issuer then decides if he wants to grant the request.

The details of the protocol are shown in Figure 2. The user first does an El-Gamal encryption of values $g_2^{\lambda_i''}$ with an ephemeral key. The issuer then chooses his own values of λ_i' and then computes the HABE keys by modifying the user's El-Gamal ciphertext. We have $\lambda_i = \lambda_i' + \lambda_i''$. The protocol is 10-move if the zero-knowledge proofs from [CDM00] are used.

$$
\begin{array}{lcl}
\underline{\mathsf{User}^{(\varphi)}(L^{(\varphi)}, PI)} & \text{Authenticated channel} & \underline{\mathsf{Issuer}(L^{(\varphi)}, SI, PI)} \\
x \xleftarrow{\$} \mathbb{Z}_p^*, \quad X \leftarrow g_2^x, & & \\
\{\lambda_i'', r_{3,i} \xleftarrow{\$} \mathbb{Z}_p\}_{i=1}^n, & & \{\lambda_i' \xleftarrow{\$} \mathbb{Z}_p\}_{i=1}^n, \quad \lambda_0 \xleftarrow{\$} \mathbb{Z}_p, \\
\left\{E_{3,i} \leftarrow g_2^{\lambda_i''} X^{r_{3,i}}\right\}_{i=1}^n, & & s \xleftarrow{\$} \mathbb{Z}_p^*, \quad \{r_{5,i} \xleftarrow{\$} \mathbb{Z}_p\}_{i=1}^n, \\
\{F_{3,i} \leftarrow g_2^{r_{3,i}}\}_{i=1}^n . & \xrightarrow{\quad L, X, \{E_{3,i}, F_{3,i}\}_{i=1}^n \quad} & D_0^{(\varphi)} \leftarrow g_2^{\frac{w+s}{\beta}}, \quad D_{0,2}^{(\varphi)} \leftarrow g_2^{\lambda_0}, \\
 & \xrightarrow{\quad \mathsf{ZKPoK}_4 \text{ (see Equation 4)} \quad} & D_{0,1}^{(\varphi)} \leftarrow g_2^s D_{0,2}^{a_{0,0}}, \\
 & & \sigma^{(\varphi)} \leftarrow \mathsf{Sign}(D_{0,2}, sgk_I), \\
 & & \left\{E_{4,i} \leftarrow g_2^{\lambda_i'} E_{3,i}\right\}_{i=1}^n, \\
 & & \left\{E_{5,i} \leftarrow g_2^s E_{4,i}^{a_{i,L_i}} X^{r_{5,i}}\right\}_{i=1}^n, \\
\left\{D_{i,2}^{(\varphi)} \leftarrow E_{4,i} F_{3,i}^{-x}\right\}_{i=1}^n, & \xleftarrow{\quad \sigma, D_0, D_{0,1}, D_{0,2}, \{E_{4,i}, E_{5,i}, F_{5,i}\}_{i=1}^n \quad} & \left\{F_{5,i} \leftarrow F_{3,i}^{a_{i,L_i}} g_2^{r_{5,i}}\right\}_{i=1}^n . \\
\left\{D_{i,1}^{(\varphi)} \leftarrow E_{5,i} F_{5,i}^{-x}\right\}_{i=1}^n, & \xleftarrow{\quad \mathsf{ZKPoK}_5 \text{ (see Equation 5)} \quad} & \\
SU^{(\varphi)} \leftarrow \left\{L, \sigma, D_0, \{D_{i,1}, D_{i,2}\}_{i=0}^n\right\}, & & \\
\textbf{return } SU^{(\varphi)}. & & \textbf{return } \{\varphi, L\}.
\end{array}
$$

Figure 2: Real-world key generation

The following proofs of knowledge are used in the key issuing protocol:

$$\mathsf{ZKPoK}_4 \left\{ \left(\boxed{x}, \{\boxed{\lambda_i''}, \boxed{r_{3,i}}\}_{i=1}^n \right) : X = g_2^{\boxed{x}} \bigwedge_{i=1}^n \left(E_{3,i} = g_2^{\boxed{\lambda_i''}} X^{\boxed{r_{3,i}}} \wedge F_{3,i} = g_2^{\boxed{r_{3,i}}} \right) \right\}, \tag{4}$$

$$\begin{aligned}
\mathsf{ZKPoK}_5 \Bigg\{ & \left(\boxed{w}, \boxed{\beta}, \boxed{s}, \left\{ \boxed{\lambda_i'}, \boxed{a_{i,L_i}} \right\}_{i=0}^n, \{\boxed{r_{5,i}}\}_{i=1}^n \right) : Y = g_{\mathrm{T}}^{\boxed{w}} \wedge B = g_1^{\boxed{\beta}} \wedge \\
& 1 = g_2^{\boxed{w}} g_2^{\boxed{s}} (D_0^{-1})^{\boxed{\beta}} \wedge D_{0,2} = g_2^{\lambda_0''} g_2^{\boxed{\lambda_0'}} \wedge D_{0,1} = g_2^{\boxed{s}} D_{0,2}^{\boxed{a_{0,0}}} \bigwedge_{i=0}^n \left(A_{i,L_i} = g_1^{\boxed{a_{i,L_i}}} \right) \\
& \bigwedge_{i=1}^n \left(E_{4,i} = g_2^{\boxed{\lambda_i'}} E_{3,i} \wedge E_{5,i} = g_2^{\boxed{s}} E_{4,i}^{\boxed{a_{i,L_i}}} X^{\boxed{r_{5,i}}} \wedge F_{5,i} = F_{3,i}^{\boxed{a_{i,L_i}}} g_2^{\boxed{r_{5,i}}} \right) \Bigg\}.
\end{aligned} \tag{5}$$

4.2.10 Query

Given a key with ACL $L = [L_0, \dots L_n]$, the user would need to compute the following if he wishes to decrypt a record [NYO08]:

$$M' \leftarrow \frac{\hat{C} \prod_{i=0}^n \mathrm{e}(C_{i,1}, D_{i,1})}{\mathrm{e}(C_0, D_0) \prod_{i=0}^n \mathrm{e}(C_{i,L_i,2}, D_{i,2})} = M g_{\mathrm{T}}^{-\sum_{i=0}^n \lambda_i \epsilon_{i,L_i}}. \tag{6}$$

The recovered message M' is equal to M if the key satisfies the record policy (i.e., if all ϵ_{i,L_i} are equal to zero). Intuitively, if the key does not satisfy the policy of the record, then the message is blinded by a random value, and the user cannot decrypt.

However, as we have seen, the database did not publish $C_{0,0,2}$. The "missing piece" $P \overset{\text{def}}{=} \mathrm{e}(C_{0,0,2}, D_{0,2}) = \mathrm{e}(C_{0,\varrho,2}, D_{0,2})^{k_\varrho^{-1}}$ will have to be computed jointly by the user and the database. We show the 10-move OT protocol to do this in Figure 3.

The user recovers the plaintext thus:

$$M' \leftarrow \frac{\hat{C} \prod_{i=0}^{n} \mathrm{e}(C_{i,1}, D_{i,1})}{P\, \mathrm{e}(C_0, D_0) \prod_{i=1}^{n} \mathrm{e}(C_{i,L_i,2}, D_{i,2})}. \tag{7}$$

The user then checks if M' was correctly recovered (i.e., if $M' = M$)[17] and returns M' if so. If not, the user returns $\perp$.

$\underline{\mathsf{U}^{(\varphi)}\left(CT^{(\psi)}, SU^{(\varphi)}, PD^{(\varrho)}, PI\right):}$	Anonymous channel	$\underline{\mathsf{DB}^{(\varrho)}(SD^{(\varrho)}, PI):}$
Blind $C_{0,\varrho,2}^{(\psi)}$ and $D_{0,2}^{(\varphi)}$: $k_c \xleftarrow{\$} \mathbb{Z}_p^*,\quad C' \leftarrow (C_{0,\varrho,2}^{(\psi)})^{k_c}$, $k_d \xleftarrow{\$} \mathbb{Z}_p^*,\quad D'' \leftarrow (D_{0,2}^{(\varphi)})^{k_d}$. Re-randomize $\sigma^{(\psi)}$ and $\sigma^{(\varphi)}$: $\sigma' \leftarrow \mathsf{SigRerand}(\sigma^{(\psi)})$, $\sigma'' \leftarrow \mathsf{SigRerand}(\sigma^{(\varphi)})$. Parse σ' and σ'' as respecively $\{Z', R', S', T', U', V', W'\}$ and $\{Z'', R'', S'', T'', U'', V'', W''\}$.	$\xrightarrow{C', S', T', V', W', D'', S'', T'', V'', W''}$	
	$\xrightarrow{\mathsf{ZKPoK}_6 \text{ (see Equation 8)}}$	$P' \leftarrow \mathrm{e}(C', D'')^{k_\varrho^{-1}}$
	$\xleftarrow{P'}$	
Unblind P': $P \leftarrow P'^{k_c^{-1} k_d^{-1}}$. Compute M' from Equation 7. If M' correct[a]: **return** M'; else **return** $\perp$.	$\xleftarrow{\mathsf{ZKPoK}_7 \text{ (see Equation 9)}}$	**return** ε.

Figure 3: Transfer protocol

[a]See Footnote 8.

The ZKPoK_6 in Figure 3 and Equation 8 is used to verify that the user possesses the signature to $C_{0,\varrho,2}^{(\psi)}$ and $D_{0,2}^{(\varphi)}$. The equations inside the ZKPoK follow straightforwardly from the verification equations of the signature scheme (cf. Appendix E.4). Both parties parse vk_ϱ as $\{g'_Z, f'_Z, g'_R, f'_U, g'_M, f'_M, A', B'\}$ and vk_I as $\{g''_Z, f''_Z, g''_R, f''_U, g''_M, f''_M, A'', B''\}$.

$$\begin{aligned}
\mathsf{ZKPoK}_6 \Big\{ \Big(\boxed{k_c^{-1}}, \boxed{Z'}, \boxed{R'}, \boxed{U'}, \boxed{k_d^{-1}}, \boxed{Z''}, \boxed{R''}, \boxed{U''} \Big) &: \\
A' &= \mathrm{e}(\boxed{Z'}, g'_Z)\, \mathrm{e}(\boxed{R'}, g'_R)\, \mathrm{e}(T', S')\, \mathrm{e}(C', g'_M)^{\boxed{k_c^{-1}}} \wedge \\
B' &= \mathrm{e}(\boxed{Z'}, f'_Z)\, \mathrm{e}(\boxed{U'}, f'_U)\, \mathrm{e}(W', V')\, \mathrm{e}(C', f'_M)^{\boxed{k_c^{-1}}} \wedge \\
A'' &= \mathrm{e}(g''_Z, \boxed{Z''})\, \mathrm{e}(g''_R, \boxed{R''})\, \mathrm{e}(S'', T'')\, \mathrm{e}(g''_M, D'')^{\boxed{k_d^{-1}}} \wedge \\
B'' &= \mathrm{e}(f''_Z, \boxed{Z''})\, \mathrm{e}(f''_U, \boxed{U''})\, \mathrm{e}(V'', W'')\, \mathrm{e}(f''_M, D'')^{\boxed{k_d^{-1}}} \Big\},
\end{aligned} \tag{8}$$

[17]See Footnote 8.

Table 1: Size of the various keys in our construction. $D \overset{\text{def}}{=} N_{\text{databases}}$ denotes the number of databases, $U \overset{\text{def}}{=} N_{\text{users}}$ the number of users, $R \overset{\text{def}}{=} N_{\text{records}}$ the total number of records, n the number of categories, and $V \overset{\text{def}}{=} \sum_{i=1}^{n} n_i$ the total number of attributes.

Item	# times	Number of elements			
	repeated	$\mathbb{Z}_p$	$\mathbb{G}_1$	$\mathbb{G}_2$	$\mathbb{G}_T$
CRS	1	–	1	4	–
Issuer secret key	1	$9 + V$	–	–	–
Issuer public key	1	–	$8 + V$	–	3
User secret key	U	–	2	$8 + 2n$	–
Database secret key	D	7	–	–	–
Database public key	D	–	1	6	2
Database record	R	1	$11 + 2n + V$	$4 + 2n$	1
Size of elements in bits[a]		κ	$\kappa + 8$	$3\kappa + 8$	6κ

[a]For a type-D curve from the PBC library with a $\leq 2^{\kappa}$ group order, elements of $\mathbb{G}_1$ and $\mathbb{G}_2$ are compressed.

$$\mathsf{ZKPoK}_7\{(\boxed{k_\varrho}) : P'^{\boxed{k_\varrho}} = \mathrm{e}(C', D'') \wedge A_{0,\varrho} = A_{0,0}^{\boxed{k_\varrho}}\}. \tag{9}$$

4.3 Theoretical Efficiency

Table 3 shows the amount of computation that the various players have to do in our scheme. Table 1 shows the size of the keys, and Table 2 shows the size of the transmitted messages during all of the protocols in our scheme.

We point out that a significant amount of work is done at the start of the scheme (setup, key and record issuing protocols). The players have to perform computation in time linear either to the number of users N_{users} times the total number of attribute $N_{\text{attributes}} \overset{\text{def}}{=} V \overset{\text{def}}{=} \sum n_i$, or computation linear to the number of records N_{records} times $N_{\text{attributes}}$, but never linear to the product of the three.[18]

For each query, the database has to do only a constant[19] amount of work, while the user has to do work linear in the number of categories. (In contrast, [CDNZ11] requires that both the database and the user do work linear in the number of attributes). In the worst case, each user will query each ciphertext, so it is very important to make sure that the query protocol is fast for the database, which has to bear most of the work.

5 Security Analysis

We prove our scheme secure in the generic bilinear group model for type-3 pairings, based on indistinguishability between the real and ideal worlds defined in Section 2.4.

The adversary may corrupt any subset of players, but must do so before the start of scheme. Furthermore, the adversary is never allowed to run more than a small constant number of ZKPoKs in parallel, as the extractor of the ZKPoK would have an exponential run-time otherwise.

[18]The only exception is the work involved in broadcasting the records' ciphertext of the database to all users. However, in this phase no cryptography is involved, so this can happen with the help of a normal (not specially secured) file server, content distribution network, or snail-mailed DVD.

[19]Constant assuming we fix the security parameter κ. If κ varies, then the run time is $\mathrm{O}(\kappa^3)$.

Table 2: Size of the various exchanged messages in our construction. $Q \overset{\text{def}}{=} N_{\text{queries}}$ denotes the number of queries, all other letters as in the table above.

Item	# times repeated	Number of elements $\mathbb{Z}_p$	$\mathbb{G}_1$	$\mathbb{G}_2$	$\mathbb{G}_T$
IssuerSetup	$U + D$	$32 + 3V$	$32 + 5V$	–	15
IssueKey	U	$47 + 12n + 3V$	$26 + 4V$	$25 + 25n$	12
DatabaseSetup	DU	26	6	22	10
IssueRecord[a]	RU	1	$11 + 2n + V$	$4 + 2n$	1
Query	Q	19	18	14	21
Size of elements in bits[b]		κ	$\kappa + 8$	$3\kappa + 8$	6κ

[a]The records can be disseminated using an untrusted server, or published on DVD.

[b]For a type-D curve from the PBC library with a $\leq 2^\kappa$ group order, elements of $\mathbb{G}_1$ and $\mathbb{G}_2$ are compressed.

Table 3: Computational costs borne by the various players in our construction. We only list the number of pairings performed, the number of exponentiations, and the number of random variables generated, since these are the major contributors to the run-time. (Multiplications, additions and inversions are "free" since these run in time quadratic to the security parameter, while exponentiations and pairings have a cubic run-time.)
D denotes the number of databases, U the number of users, R the total number of records and R_D the number of records issued by a particular database, Q_U the number of queries performed by a particular user and Q_D the number of queried processed by a particular database, n the number of categories, and V the total number of attributes. Let m be the number of attributes that are not in the ciphertext policy of a given record.

Item	# times repeated	Computational cost Expon. $\mathbb{G}_1$	Expon. $\mathbb{G}_2$	Expon. $\mathbb{G}_T$	Pairings	Random $\mathbb{Z}_p$[a]
• Issuer •						
IssuerSetup	1	$8 + V$	2	1	2	$11 + V$
CheckIssuerKey	$U + D$	$42 + 7V$	–	21	2	$21 + 2V$
IssueKey	U	$16 + 7n$	$49 + 58n$	7	–	$25 + 12n$
• Database •						
CheckIssuerKey	1	$54 + 9V$	–	27	2	$21 + 2V$
DatabaseSetup	1	3	6	–	2	9
CheckDatabaseKey	U	7	28	14	2	17
IssueRecord	R_D	$16 + 4n + V$	$10 + 4n$	1	–	$8 + 2n + m$
Query	Q_D	17	9	45	41	24
• User •						
CheckIssuerKey	1	$54 + 9V$	–	27	2	$21 + 2V$
IssueKey	1	$18 + 9n$	$46 + 61n$	9	8	$20 + 12n$
CheckDatabaseKey	D	9	36	18	2	17
CheckRecord	R	–	4	–	$26 + 8n$	–
Query	Q_U	27	22	38	$35 + 2n$	34
• Run time[b] per operation •		1.75 ms	15.6 ms	4.1 ms	13.6 ms	1.1 ms

[a]Generating random values in $\mathbb{Z}_p$ is linear to the size of the security parameter but takes a non-negligible amount of time.

[b]The run-time estimates are given for a type-D curve from the PBC library with 158-bit group order on our computer (Intel T2600, 2.16GHz). For random elements in $\mathbb{G}_1$ or $\mathbb{G}_2$ we first need to generate a random element of $\mathbb{Z}_p$, then do an exponentiation, for a time of around 2.8 ms.

Theorem 1 If the HABE scheme is (match-concealing) secure, the SXDH assumption holds, and the $\max(N_{\text{users}}, N_{\text{records}})$-SFP assumption holds in the chosen bilinear group, then:[20] For all subsets of corrupted parties, and for all PPT algorithms $\mathcal{A}$ describing the behaviour of the corrupted parties, we can construct a simulator $\mathcal{S}$—which has rewindable black-box access to $\mathcal{A}$ and may dictate the CRS to $\mathcal{A}$—such that the advantage of all environments $\mathcal{E}$ in distinguishing between the real and the ideal world is negligible. That is:

$$\forall \mathcal{A} : \exists \mathcal{S} : \forall \mathcal{E} : \mathbf{Adv}^{\text{HABE-OT}}_{\mathcal{E},\mathcal{A}} \overset{\text{def}}{=} \left| \Pr\left[\text{Out}_{\mathcal{E}}(\mathcal{E} \overset{\text{Real}}{\longleftrightarrow} \mathcal{A}) \overset{\$?}{=} 1\right] - \Pr\left[\text{Out}_{\mathcal{E}}(\mathcal{E} \overset{\text{Ideal}}{\longleftrightarrow} \mathcal{S}) \overset{\$?}{=} 1\right]\right| = \mathit{negl}.$$

Corollary 2 The scheme presented in Section 4.2 satisfies all security properties described in Section 2.5.

There are five main cases to consider, which are summarized in Table 4. We do not consider the other cases in our proof: we can construct a simulator $\mathcal{S}$ for them by combining the strategies used for the main cases.

Table 4: Main cases to consider for the security proof. An "X" means that some of the players in the respective category are corrupted, while a hyphen "–" means that all players in the category are honest.

Case	Issuer	Database	User	Comments
1	–	–	–	Guaranteed by completeness, see Section 5.1.
2	X	–	–	Subsumed by case 5.
3	–	X	–	See Lemma 3, proved in Section 5.2.
4	–	–	X	See Lemma 4, proved in Section 5.3.
5	X	X	–	See Lemma 5, proved in Section 5.4.

Lemma 3 If the SXDH assumption holds, then: For all $\mathcal{A}$ in which any number of databases are corrupt, we can always construct $\mathcal{S}(\mathcal{A})$ controlling the corrupt parties in the ideal world such that all $\mathcal{E}$ have at most a negligible advantage in the distinguishing game.

Lemma 4 If the HABE scheme we use is (match-concealing) secure, the SXDH assumption holds, and the $\max(N_{\text{users}}, N_{\text{records}})$-SFP assumption holds, then: For all $\mathcal{A}$ in which any number of users are corrupt, we can always construct $\mathcal{S}(\mathcal{A})$ such that all $\mathcal{E}$ have at most a negligible advantage in the distinguishing game.

Lemma 5 If the SXDH assumption holds: For all $\mathcal{A}$ in which the issuer and any number of databases are corrupt, we can always construct $\mathcal{S}(\mathcal{A})$ such that all $\mathcal{E}$ have at most a negligible advantage in the distinguishing game.

Proof strategy For cases 3–5, we are going to define a sequence of games, as described by Shoup [Sho04]. In the zeroth game, everything is distributed as in the real world, whereas in the last game everything is distributed as in the ideal world. By the piling-up lemma, the advantage of $\mathcal{E}$ is less than the sum of the advantages in distinguishing between Games i and Game $i+1$. We are going to prove that $\mathcal{E}$ only has negligible advantage in distinguishing between two consecutive games, based either on a reduction to a hard cryptographic problem,

[20]Since the HABE scheme was proven secure only in the generic bilinear group model of type-3 pairings, one could instead say that our construction is secure in the generic bilinear group model.

or by "failure events" happening with negligible probability. As long as the number of games is polynomial w.r.t. the security parameter, the total advantage of $\mathcal{E}$ is negligible.

We must stress that in all intermediate games, the simulator $\mathcal{S}$ has complete access to all honest parties. We make use of this fact in the security proof. Of course, this is not the case in the ideal world, so by the last game, $\mathcal{S}$ doesn't need that extra information anymore.

5.1 Completeness

Completeness for the users and the databases follows from the completeness of the HABE scheme, the signature scheme, the GS and zero-knowledge proofs.

Real-world escrow To prove completeness for the issuer, we also need to show how the escrow functionality works. The issuer can recover the plaintext from a given ciphertext $\hat{C}, C_0$ by computing:

$$M' \leftarrow \hat{C}\, \mathrm{e}(C_0, g_2)^{-w\beta^{-1}} = M g_{\mathrm{T}}^{wr}\, \mathrm{e}(g_1^{\beta r}, g_2)^{-w\beta^{-1}} = M.$$

If M is used to generate a symmetric key, the issuer needs to check that it allows a correct decryption of the symmetric ciphertext. In the ideal world, the case where the issuer can't decrypt even with the correct M can be simulated by having a ciphertext policy that permits no one to decrypt (e.g., setting $W_1 = \emptyset$).

The issuer can determine if a given ciphertext component $C_{i,t,2}$ ($i \neq 0$) was computed correctly or not (and thus recover the whole ciphertext policy):

$$C_{i,1}^{a_{i,t}} \stackrel{?}{=} C_{i,t,2}.$$

Note that for $i = 0$, the perfect soundness of the NIZK_3 proof guarantees that $C_{0,\varrho,2}$ is well-formed and that $r = \sum r_i$.

5.2 Corrupted Database

Honest databases don't communicate with one another, they share no common secret, and furthermore they can never violate the user's privacy: dishonest databases thus gain nothing by colluding. We can therefore, without loss of generality, handle the case of only one dishonest database.

Game 1 $\mathcal{S}_1$ generates a hiding CRS and gives it to $\mathcal{A}$. (We do this for compatibility with the dishonest user case.) The difference between this game and the real world is negligible based on the SXDH assumption [GS08].

Game 2 $\mathcal{S}_2$ runs like $\mathcal{S}_1$ except that it runs an honest issuer $\mathcal{S}_I$ and gives that simulated issuer's public key to $\mathcal{A}$. $\mathcal{S}_I$ also generates a user key for $\mathcal{S}_2$. The difference between the two games is zero.

Game 3 $\mathcal{S}_3$ runs like $\mathcal{S}_2$ except that when $\mathcal{A}$ publishes an encrypted record with a correct GS proof, $\mathcal{S}_2$ runs the escrow functionality to extract the message and the ciphertext policy of that record. The difference between the two games is zero, because of the perfect soundness of GS proofs and the perfect completeness of the escrow functionality.

Game 4 $\mathcal{S}_4$ runs like $\mathcal{S}_3$ except that when a user U queries for a missing piece, $\mathcal{S}_3$ queries a random piece from $\mathcal{A}$. The difference between the two games is zero, because of the perfect zero-knowledge property.

Game 5 $\mathcal{S}_5$ runs like $\mathcal{S}_4$ except that it uses its rewindable black-box access of $\mathcal{A}$ to extract the witnesses from ZKPoK_7. $\mathcal{S}_5$ aborts if it fails to extract. The difference between the two games is $2^{-\kappa} N_{\text{queries}}$.

Game 6 $\mathcal{S}_6$ runs exactly like $\mathcal{S}_5$ except that it aborts if the user recovers the correct message despite the fact that his key does not satisfy the policy. This happens only when

$$\exists j, 0 \leq j \leq n: \quad \epsilon_{j,L_j} \neq 0 \pmod{p} \quad \wedge \quad -\sum_{i=0}^{n} \lambda_i \epsilon_{i,L_i} = 0 \pmod{p}.$$

Since the values λ_i are random, and the database never sees these values, this failure event happens with probability p^{-1}, which is negligible.

Construction We now construct a simulator $\mathcal{S}$ which incorporates all steps from the previous games. $\mathcal{S}$ relays all messages he gets from $\mathcal{E}$ to $\mathcal{A}$, and all return values from $\mathcal{A}$ to $\mathcal{E}$.

$\mathcal{S}$ generates a hiding CRS and gives it to $\mathcal{A}$.

$\mathcal{S}$ runs like the honest issuer to generate the issuer's public key. It also generates a key k for itself.

When $\mathcal{S}$ receives an encrypted record from $\mathcal{A}$, it uses the escrow functionality to recover the plaintext and the ciphertext policy. It then sends these to T.

When $\mathcal{S}$ receives a query message from T, $\mathcal{S}$ chooses a random record, and its own key k and queries $\mathcal{A}$ with these. If $\mathcal{A}$ is cooperative in the query protocol, $\mathcal{S}$ sends $c = 1$ to T. Else $\mathcal{S}$ sends $c = 0$ to T (remember that it aborts if $\mathcal{A}$ cheated in ZKPoK_7).

5.3 Corrupted User

Game 1 (Offline guessing) Users are never issued enough attributes to decrypt messages from the database by themselves. The transition between Game 1 and the real world is based on the failure event (in the sense of [Sho04]) that the adversary recovers the plaintext of that record without having queried the database for that record, or manages to get more information about the policy of a record than what he could deduce by interacting with the database. The security proof of the HABE scheme guarantees that the probability of this event is negligible, thus $\mathcal{E}$ cannot get non-negligible advantage from this game.

Game 2 $\mathcal{S}_2$ generates a hiding CRS with trapdoor. $\mathcal{S}_2$ uses that trapdoor to fake all GS proofs. The difference between this game and the real world is the different distribution of the CRS, which can only be distinguished with negligible advantage based on the SXDH assumption [GS08].

Game 3 $\mathcal{S}_3$ runs like $\mathcal{S}_2$ except that each time $\mathcal{A}$ asks for a key, $\mathcal{S}_3$ extracts x from ZKPoK_4. The difference between the two games is the knowledge error of the ZKPoK, i.e., $2^{-\kappa} N_{\text{users}}$.

Game 4 $\mathcal{S}_4$ runs like $\mathcal{S}_3$ except that it chooses the values of D_0, $\{D_{i,1}, D_{i,2}\}_{i=0}^{n}$ independently from the λ'' values of the user (but still according to the rules of the HABE scheme). $\mathcal{S}_4$ can compute the correct values of $\{E_{4,i}, E_{5,i}\}_{i=1}^{n}$ with the help of x. $\mathcal{S}_4$ then fakes ZKPoK_5 using the rewindable black-box access. The distribution is exactly the same between the two games and therefore the difference of advantage is zero.

Game 5 $\mathcal{S}_5$ runs like $\mathcal{S}_4$ except that it extracts k_c and k_d from ZKPoK_6 each time $\mathcal{A}$ makes a query. This allows $\mathcal{S}$ to unblind C' and D'' and therefore to identify the user φ and the record ψ. The difference between the two games is the knowledge error of the ZKPoK, i.e., $2^{-\kappa}N_{\text{queries}}$.

Game 6 $\mathcal{S}_6$ runs like $\mathcal{S}_5$ except that if the user key φ or the record ψ was never issued, it aborts. This means that $\mathcal{A}$ managed to fake a signature. The difference between the two games is the advantage of the $(\max(N_{\text{users}}, N_{\text{records}})) - SFP$ assumption.

Game 7 $\mathcal{S}_7$ runs like $\mathcal{S}_6$ except that each time that $\mathcal{A}$ queries for a record ψ on behalf of user φ, $\mathcal{S}_7$ computes P' as follows:

$$P' \leftarrow \left(\frac{\hat{C}^{(\psi)} \prod_{i=0}^{n} \mathrm{e}(C_{i,1}^{(\psi)}, D_{i,1}^{(\varphi)})}{M^{(\psi)}\, \mathrm{e}(C_0^{(\psi)}, D_0^{(\varphi)}) \prod_{i=1}^{n} \mathrm{e}(Q_{i,L_i^{(\varphi)},2}^{(\psi)}, D_{i,2}^{(\varphi)})} \right)^{k_c k_d}, \tag{10}$$

$$\text{where } \forall i \in \mathbb{N}_{n+1}^{*} \forall t \in \mathbb{N}_{n_i} : Q_{i,t,2}^{(\psi)} \begin{cases} \leftarrow C_{i,t,2}^{(\psi)} & \text{if } t \in W_i^{(\psi)} \text{ (component in policy);} \\ \xleftarrow{\$} \mathbb{G}_1 & \text{if } t \notin W_i^{(\psi)} \text{ (component not in policy).} \end{cases}$$

We would like to point out that P' is computed correctly (i.e., as in Game 6) if the key φ satisfies the policy of record ψ (i.e., $\forall i \geq 1 : L_i^{(\varphi)} \in W_i^{(\psi)}$).

To handle the case where $\mathcal{A}$ queries several times for the same tuple (ψ, φ), $\mathcal{S}$ computes the value $P''_{\varphi,\psi} = P'^{k_c^{-1} k_d^{-1}}$ in the first query (where P' is as above). In subsequent queries $\mathcal{S}$ replies with $P''^{k_c k_d}_{\varphi,\psi}$, so that the user recovers the same value of P' after unblinding. For the rest of the discussion we will assume, without loss of generality, that $\mathcal{A}$ never queries twice for the same tuple.

This game is interesting only for the security proof, since we "undo" the changes this game makes in the next game. Game 7 and Game 8 are in fact identical if there is only one corrupted user.

Lemma 7 $\mathcal{E}$ has only negligible advantage in distinguishing between Game 6 and Game 7 based on the security of the HABE scheme.

See the proof in Section 5.5.

Game 8 $\mathcal{S}_8$ runs like $\mathcal{S}_6$ (and like $\mathcal{S}_7$) except that each time that $\mathcal{A}$ queries for a record ψ on behalf of user φ, $\mathcal{S}_8$ computes P' as follows:

If the key φ satisfies the policy of the record, $\mathcal{S}_8$ computes P' like in Games 6 and 7.

Else, $\mathcal{S}_8$ computes P' randomly: $P' \xleftarrow{\$} \mathbb{G}_{\mathrm{T}}$.

Lemma 8 $\mathcal{E}$ has only a negligible advantage in distinguishing between Game 7 and Game 8 based on the DDH assumption in $\mathbb{G}_2$.

See the proof in Section 5.6.

Game 9 $\mathcal{S}_9$ runs like $\mathcal{S}_8$ except that each time that the honest database issues a record, $\mathcal{S}_9$ computes the ciphertext for a random plaintext $\tilde{M}$ under a random ciphertext policy $\tilde{W}$ (but with $\tilde{W}_0 \not\ni 0$). $\mathcal{S}_9$ publishes the resulting ciphertext.

Each time that $\mathcal{A}$ queries for a record ψ on behalf of user φ, $\mathcal{S}_9$ checks if the key satisfies the policy of the record.

If the key φ does not satisfy the policy of record ψ, $\mathcal{S}_9$ computes P' randomly: $P' \xleftarrow{\$} \mathbb{G}_{\mathrm{T}}$. Else if the key φ satisfies the policy of record ψ, $\mathcal{S}_9$ computes P' thus:

$$P' \leftarrow \left(\frac{\hat{C}^{(\psi)} \prod_{i=0}^{n} \mathrm{e}(C_{i,1}^{(\psi)}, D_{i,1}^{(\varphi)})}{M^{(\psi)}\, \mathrm{e}(C_0^{(\psi)}, D_0^{(\varphi)}) \prod_{i=1}^{n} \mathrm{e}(C_{i,L_i^{(\varphi)},2}^{(\psi)}, D_{i,2}^{(\varphi)})} \right)^{k_c k_d} . \qquad (11)$$

In this case $P = P'^{k_c^{-1} k_d^{-1}}$ is such that when plugged in Equation 7, the latter returns the "real" plaintext $M^{(\psi)}$.

Lemma 9 $\mathcal{E}$ has only a negligible advantage in distinguishing between Game 8 and Game 9 based on the security of the HABE scheme.

See the proof in Section 5.7.

Construction We now construct a simulator $\mathcal{S}$ which incorporates all steps from the previous games. $\mathcal{S}$ relays all messages he gets from $\mathcal{E}$ to $\mathcal{A}$ and all return values from $\mathcal{A}$ to $\mathcal{E}$.

$\mathcal{S}$ runs an honest issuer and a number of honest databases to interact with $\mathcal{A}$.

$\mathcal{S}$ generates a hiding CRS with trapdoor and gives it to $\mathcal{A}$.

When $\mathcal{A}$ asks for a key, $\mathcal{S}$ relays the query to T. If it gets back the response $b = 1$ from T, $\mathcal{S}$ runs the issuer as in Game 4 to compute the key for $\mathcal{A}$.

When $\mathcal{S}$ receives a record issue notification from T, it selects a random plaintext $\tilde{M}$ and a random policy $\tilde{W}$ (where $\tilde{W}_0 \not\ni 0$). $\mathcal{S}$ publishes the resulting ciphertext.

When $\mathcal{A}$ performs a query, $\mathcal{S}$ uses the rewindable black-box access to extract k_c and k_d from $\mathcal{A}$, and then unblinds the values C' and D'' to identify φ and ψ ($\mathcal{S}$ aborts if the extraction fails, or if no signature on $C'^{k_c^{-1}}$ or $D''^{k_d^{-1}}$ was ever issued). $\mathcal{S}$ then queries T with key φ for the record ψ. $\mathcal{S}$ computes P' as described in Game 9 (i.e., random if T's reply was $\perp$, else according to Equation 11), and fakes ZKPoK_7 using its rewindable black-box access.

5.4 Corrupted Issuer + Some Corrupted Databases

If the issuer is corrupted, the only thing we have to worry about is the privacy in the query protocol of all honest users. We don't need to care about honest databases, since their records can be decrypted anyway by the issuer thanks to the escrow functionality.

We assume that all users are present from the start, and that all users are able to verify ZKPoK_1 before the database is set up.

Game 1 $\mathcal{S}_1$ generates a hiding CRS and gives it to $\mathcal{A}$. (We do this for compatibility with the dishonest user case.) The difference between this game and the real world is negligible based on the SXDH assumption [GS08].

Game 2 $\mathcal{S}_2$ runs like $\mathcal{S}_1$ except that it uses its rewindable black-box access of $\mathcal{A}$ to extract the witnesses from ZKPoK_1. $\mathcal{S}_2$ aborts if it fails to extract. The difference between the two games is $2^{-\kappa} N_{\text{users}}$.

Game 3 $\mathcal{S}_3$ runs like $\mathcal{S}_2$ except that when $\mathcal{A}$ publishes an encrypted record $\mathcal{S}_2$ extracts the message and the ciphertext policy of that record using the escrow functionality. The difference between the two games is zero.

Game 4 $\mathcal{S}_4$ runs like $\mathcal{S}_3$ except that it uses its rewindable black-box access of $\mathcal{A}$ to extract k_{ϱ} from ZKPoK_7. $\mathcal{S}_4$ aborts if it fails to extract. The difference between the two games is $2^{-\kappa}N_{\text{queries}}$.

Game 5 $\mathcal{S}_5$ runs like $\mathcal{S}_4$ except that when an honest user U queries for a missing piece, $\mathcal{S}_5$ queries a random piece from $\mathcal{A}$. The difference between the two games is zero, because of the perfect zero-knowledge property.

Game 6 $\mathcal{S}_6$ runs like $\mathcal{S}_5$ except that when an honest user U asks for a key, $\mathcal{S}_6$ extracts the witnesses from ZKPoK_5. If it fails to extract, it aborts. The difference between the two games is $2^{-\kappa}N_{\text{users}}$.

Game 7 $\mathcal{S}_7$ runs exactly like $\mathcal{S}_6$ except that it aborts if the user recovers the correct message despite the fact that his key does not satisfy the policy. This happens only when

$$\exists j, 0 \leq j \leq n: \quad \epsilon_{j,L_j} \neq 0 \pmod{p} \quad \wedge \quad -\sum_{i=0}^{n} \lambda_i \epsilon_{i,L_i} = 0 \pmod{p}.$$

There are two possibilities why this could happen: Either $\mathcal{A}$ was completely unaware of the λ_i values when generating the record (for a given query the success probability is p^{-1} in this case, which is negligible). The other possibility is that $\mathcal{A}$ somehow recovered λ_i and maliciously adapted the ϵ_{i,L_i} values. We show that ths can only happen with negligible probability:

Lemma 10 $\mathcal{E}$ has only a negligible advantage in distinguishing between Game 6 and Game 7 based on the semantic security of El-Gamal in $\mathbb{G}_2$.

See the proof in Section 5.8.

Construction We now construct a simulator $\mathcal{S}$ which incorporates all steps from the previous games. $\mathcal{S}$ relays all messages he gets from $\mathcal{E}$ to $\mathcal{A}$, and all return values from $\mathcal{A}$ to $\mathcal{E}$.

$\mathcal{S}$ runs a number of honest users and honest databases to interact with $\mathcal{A}$.

$\mathcal{S}$ generates a hiding CRS and gives it to $\mathcal{A}$.

When $\mathcal{S}$ receives the issuer public key of $\mathcal{A}$, it runs the extractor of ZKPoK_5 to recover the secret key of the issuer.

When $\mathcal{S}$ receives an encrypted record from $\mathcal{A}$, it uses the escrow functionality to recover the plaintext and the ciphertext policy. It then sends these to T.

When $\mathcal{S}$ receives a record issue notification from T, it runs the ideal world escrow to recover the plaintext and the ciphertext policy. It then asks one of the honest databases in the real world to generate a record with the same plaintext and policy.

When $\mathcal{S}$ receives a request to generate a key from T, it asks $\mathcal{A}$ to generate a key with the same permissions. If $\mathcal{A}$ is cooperative, then $\mathcal{S}$ returns $b = 1$ to T, else it returns $b = 0$ to T (if it failed to extract the witnesses from the ZKPoK, then $\mathcal{S}$ aborts).

When $\mathcal{S}$ receives a request for a key from T, it runs one of its users to query a key with the same policy to $\mathcal{A}$. If $\mathcal{A}$ is cooperative then $\mathcal{S}$ saves the key and sends $b = 1$ to T. Else $\mathcal{S}$ sends $b = 0$ to T (it aborts if it fails to extract the witness).

When $\mathcal{S}$ receives a query message from T, $\mathcal{S}$ chooses a random record, and a random user for which it knows the key and queries $\mathcal{A}$ with these. If $\mathcal{A}$ is cooperative in the query protocol, $\mathcal{S}$ sends $c = 1$ to T. Else $\mathcal{S}$ sends $c = 0$ to T (remember that it aborts if $\mathcal{A}$ cheated in ZKPoK_7).

When $\mathcal{A}$ queries for a message from an honest database, $\mathcal{S}$ simply asks T for a random record (the ideal database doesn't get to see neither the user index nor the record index).

5.5 Proof of Lemma 7

If there exists a PPT algorithm $\mathcal{D}$ which distinguishes between Game 6 and Game 7 with non-negligible advantage, then we can construct a PPT algorithm $\mathcal{S}(\mathcal{D})$ which has rewindable black box access to $\mathcal{D}$ and which plays the HABE game with non-negligible advantage.

We defines the sequence of games: Game 6-0 to Game 6-N_{records}. In game 6-ω, the value P' for all records $\psi \leq \omega$ is computed as in Game 7, and the value P' for all records $\psi > \omega$ is computed as in Game 6. Clearly, Game 6-0 is exactly Game 6, and Game 6-N_{records} is exactly Game 7. The existence of $\mathcal{D}$ implies the existence of $\mathcal{D}_\omega$ (for some ω) which can distinguish between Game 6-$(\omega - 1)$ and Game 6-ω with an advantage N_{records}^{-1} that of $\mathcal{D}$.

The challenger starts by generating the issuer key, which he transmits to $\mathcal{S}(\mathcal{D}_\omega)$. $\mathcal{S}(\mathcal{D}_\omega)$ relays it to $\mathcal{D}_\omega$.

$\mathcal{S}(\mathcal{D}_\omega)$ runs $\mathcal{D}_\omega$, who may request a key with attribute list $\{L_i\}$ where $L_0 = 0$. $\mathcal{S}(\mathcal{D}_\omega)$ relays the request to the challenger and the response back to $\mathcal{D}_\omega$. This can be repeated polynomially many times.

When $\mathcal{D}_\omega$ wants to encrypt a record $\psi \neq \omega$ (not the challenge record), with plaintext $M^{(\psi)}$ and policy $W^{(\psi)}$ (with $W_0^{(\psi)} = \{\varrho\}$),[21] $\mathcal{S}(\mathcal{D}_\omega)$ computes the ciphertext under the policy $W^{(\psi)\prime} = \{\{0, \varrho\}, W_1^{(\psi)}, W_2^{(\psi)}, \ldots, W_n^{(\psi)}\}$ (which includes the "issuer" attribute), and sends every component except $C_{0,0,2}^{(\psi)}$ (the one corresponding to the "issuer" attribute) to $\mathcal{D}_\omega$.

When $\mathcal{D}_\omega$ wants to encrypt the challenge record $\psi = \omega$ with plaintext $M^{(\omega)}$ and policy $W^{(\omega)}$ (with $W_0^{(\omega)} = \{\varrho\}$), $\mathcal{S}(\mathcal{D}_\omega)$ generates the following challenge plaintexts and policies: $M^{(0)} \leftarrow M^{(\omega)}$ and $W^{(0)} \leftarrow \{\{\varrho\}, \mathbb{N}_{n_1}, \mathbb{N}_{n_2}, \ldots, \mathbb{N}_{n_n}\}$ (the most liberal policy), as well as $M^{(1)} \leftarrow M^{(\omega)}$ and $W^{(1)} \leftarrow W^{(\omega)}$ (these are acceptable, since all keys issued will satisfy neither of $W^{(0)}$ and $W^{(1)}$). These are then sent to the challenger. The challenger flips a bit $b \xleftarrow{\$} \mathbb{N}_2$ and encrypts $M^{(b)}$ under policy $W^{(b)}$ yielding the ciphertext:

$$\{\tilde{C}^{(b)}, C_0^{(b)}, C_{0,1}^{(b)}, \{C_{0,0,2}^{(b)}, C_{0,\varrho,2}^{(b)}\}, \{C_{i,1}^{(b)}, \{\underline{Q_{i,t,2}^{(b)}}\}_{t=0}^{n_i-1}\}_{i=1}^{n}\}.$$

$\mathcal{S}(\mathcal{D}_\omega)$ now computes $C_{i,t,2}^{(b)}$ as follows:

$$\begin{aligned} &\forall i \in \mathbb{N}_{n+1}^* : \forall t \in W_i^{(\omega)} : && C_{i,t,2}^{(b)} \leftarrow Q_{i,t,2}^{(b)}; \\ &\forall i \in \mathbb{N}_{n+1}^* : \forall t \in \mathbb{N}_{n_i} \setminus W_i^{(\omega)} : && C_{i,t,2}^{(b)} \xleftarrow{\$} \mathbb{G}_1. \end{aligned}$$

That is, $\mathcal{S}(\mathcal{D}_\omega)$ re-randomizes all ciphertext components that are not in the policy. $\mathcal{S}(\mathcal{D}_\omega)$ now sends the following modified ciphertext to $\mathcal{D}_\omega$:

$$\{\tilde{C}^{(b)}, C_0^{(b)}, C_{0,1}^{(b)}, \{C_{0,\varrho,2}^{(b)}\}, \{C_{i,1}^{(b)}, \{\underline{C_{i,t,2}^{(b)}}\}_{t=0}^{n_i-1}\}_{i=1}^{n}\}.$$

In the view of $\mathcal{D}_\omega$, this challenge ciphertext is distributed exactly as if it was encrypted under the ciphertext policy $W^{(\omega)}$.

$\mathcal{S}(\mathcal{D}_\omega)$ runs $\mathcal{D}_\omega$, who may now ask for more keys (polynomially many times).

[21] Note that $\mathcal{S}_7$ can intercept the "IssueRecord" message from the environment $\mathcal{E}$ in the hybrid games.

To process queries by $\mathcal{D}_\omega$, $\mathcal{S}(\mathcal{D}_\omega)$ first extracts from ZKPoK_6 to recover the key index φ and the record index ψ. $\mathcal{S}(\mathcal{D}_\omega)$ will have no problem in processing queries for the non-challenge records $\psi \neq \omega$ to return P' of the requisite distribution. For the challenge record $\psi = \omega$, $\mathcal{S}(\mathcal{D}_\omega)$ computes P' as follows, and sends the result back to $\mathcal{D}_\omega$:

$$P' \leftarrow \left(\frac{\hat{C}^{(b)} \prod_{i=0}^{n} \mathrm{e}(C_{i,1}^{(b)}, D_{i,1}^{(\varphi)})}{M^{(0)} \mathrm{e}(C_0^{(b)}, D_0^{(\varphi)}) \prod_{i=1}^{n} \mathrm{e}(Q_{i,L_i^{(\varphi)},2}^{(b)}, D_{i,2}^{(\varphi)})} \right)^{k_c k_d} . \tag{12}$$

If $b = 0$ or if the key φ satisfies the policy of record ω, then all values of $Q_{i,t,2}$ in the above equation are non-random, we therefore have that $P' = \mathrm{e}(C_{0,\varrho,2}^{(\omega)}, D_{0,2}^{(\varphi)})^{k_\varrho^{-1} k_c k_d}$ and thus distributed exactly as in Game 6. Else P' is distributed exactly in Game 7 (it contains random values that $\mathcal{D}_\omega$ doesn't have).

The query phase may be repeated polynomially many times.

$\mathcal{S}(\mathcal{D}_\omega)$ runs $\mathcal{D}_\omega$, who now finally outputs a guess of b. $\mathcal{S}(\mathcal{D}_\omega)$ sends b to the challenger. $\mathcal{S}(\mathcal{D}_\omega)$ has the same advantage in the HABE security game, as $\mathcal{D}_\omega$ has in distinguishing between Game 6-$(\omega - 1)$ and Game 6-ω.

$$\mathbf{Adv}_{\mathcal{D}}^{\text{Game 6–7}} \leq \max_{\omega} \left(N_{\text{records}} \cdot \mathbf{Adv}_{\mathcal{D}_\omega}^{\text{Game 6-}(\omega\,-\,1)\text{–6-}\omega} \right) \leq \max_{\omega} \left(N_{\text{records}} \cdot \mathbf{Adv}_{\mathcal{S}(\mathcal{D}_\omega)}^{\text{HABE}} \right) = \textit{negl}.$$

5.6 Proof of Lemma 8

Difference between Game 7 and 8 In Game 7, P' for a given ψ and variable φ is *not* distributed *uniformly* at random (informally, it's because there isn't enough “randomness” in P' if you have multiple keys), while it is random and *independant* in Game 8. (Of course if there is a single key, then both Games 8 and 7 are the same. Also if you fix φ and vary ψ, the P' values will be uniformly random).

The proof We show that if a PPT algorithm $\mathcal{D}$ distinguishes between Game 7 and Game 8, we can construct an algorithm $\mathcal{S}(\mathcal{D})$ which plays the DDH game in $\mathbb{G}_2$ with non-negligible advantage.

For this we make use of $(N_{\text{users}} N_{\text{records}} + 1)$ hybrid games Game 7-0-0 to Game 7-N_{users}-N_{records},[22] where in Game 7-ϕ-ω the values of P' are computed as in Game 8 for all keys φ and records ψ satisfying $(\varphi, \psi) \leq (\phi, \omega)$ (lexicographical comparison) and computed as in Game 7 for $(\varphi, \psi) > (\phi, \omega)$. Clearly Game 7-0-0 is exactly Game 7, and Game 7-N_{users}-N_{records} is exactly Game 8.

The existence of $\mathcal{D}$ implies the existence of $\mathcal{D}_{\phi\omega}$ (for some ϕ and some ω) which can distinguish between Game 7-ϕ-ω and the lexicographically preceding game with advantage $\frac{1}{N_{\text{users}} N_{\text{records}}}$ that of $\mathcal{D}$. We now show how to construct an algorithm $\mathcal{S}(\mathcal{D}_{\phi\omega})$ which plays the DDH game with advantage at least $\frac{1}{N_{\text{attributes}}} \overset{\text{def}}{=} \frac{1}{\sum_{i=1}^{n} n_i}$ that of $\mathcal{D}_{\phi\omega}$, which is still a non-negligible advantage:

$\mathcal{S}(\mathcal{D}_{\phi\omega})$ begins by choosing a category $j \overset{\$}{\leftarrow} \mathbb{N}_{n+1}^*$ and an attribute $j' \overset{\$}{\leftarrow} \mathbb{N}_{n_i}$ from that category at random. $\mathcal{S}(\mathcal{D}_{\phi\omega})$ receives a DDH challenge $\left(g_2^\gamma, g_2^\delta, g_2^z\right)$ where z is either $\gamma\delta$ or random. $\mathcal{S}(\mathcal{D}_{\phi\omega})$ generates the public key of the issuer and gives them to $\mathcal{D}_{\phi\omega}$.

$\mathcal{S}(\mathcal{D}_{\phi\omega})$ runs $\mathcal{D}_{\phi\omega}$ for the key issuing phase. $\mathcal{D}_{\phi\omega}$ may choose to be issued up to a polynomial number of keys (under the restriction that $L_0 = 0$). When $\mathcal{S}(\mathcal{D}_{\phi\omega})$ issues the key $\varphi = \phi$: First it checks that $j' = L_j^{(\phi)}$ (else $\mathcal{S}(\mathcal{D}_{\phi\omega})$ aborts and takes a random guess), then it sets

[22] To simplify notation somewhat, we will ignore the games where no progress is made.

$\lambda_j^{(\phi)} = \gamma$ (without being able to compute that value), and finally computes $D_{j,1}^{(\phi)} \leftarrow g_2^s (g_2^\gamma)^{a_{j,j'}}$ and $D_{j,2}^{(\phi)} \leftarrow g_2^\gamma$. All other components of the key are computed honestly. For all other keys $\varphi \neq \phi$, $\mathcal{S}(\mathcal{D}_i)$ computes $\lambda_j^{(\varphi)} \xleftarrow{\$} \mathbb{Z}_p^*$, $D_{j,2}^{(\varphi)} \leftarrow g_2^{\lambda_j^{(\varphi)}}$ and $D_{j,1}^{(\varphi)} \leftarrow g_2^s (D_{j,2}^{(\varphi)})^{a_{j,L_j^{(\varphi)}}}$ (so that it knows the discrete logarithm of $D_{j,2}^{(\varphi)}$), and all other key components honestly. $\mathcal{S}(\mathcal{D}_{\phi\omega})$ fakes ZKPoK_5 using its black-box access to $\mathcal{D}_{\phi\omega}$.

$\mathcal{S}(\mathcal{D}_{\phi\omega})$ runs $\mathcal{D}_{\phi\omega}$, who chooses a plaintext $M^{(\omega)}$ and ciphertext policy $W^{(\omega)}$. $\mathcal{S}(\mathcal{D}_{\phi\omega})$ aborts if $j' \in W_j^{(\omega)}$.

$\mathcal{S}(\mathcal{D}_{\phi\omega})$ computes the ciphertext:

$$\{\tilde{C}^{(\omega)}, C_0^{(\omega)}, C_{0,1}^{(\omega)}, \{C_{0,0,2}^{(\omega)}, C_{0,\varrho,2}^{(\omega)}\}, \{C_{i,1}^{(\omega)}, \{\underline{Q_{i,t,2}^{(\omega)}}\}_{t=0}^{n_i-1}\}_{i=1}^n\}.$$

however instead of computing $Q_{j,j',2}^{(\omega)}$ honestly, it embeds part of the DDH challenge in that value: it sets $Q_{j,j',2}^{(\omega)} = g_1^\delta$ (without being able to compute this value since it only knowns g_2^δ). All other ciphertext components $Q_{i,t,2}^{(\omega)}$ are computed honestly. Like in the last game however, $\mathcal{S}(\mathcal{D}_{\phi\omega})$ re-randomizes all ciphertext components which are not part of the policy:

$$\begin{aligned} &\forall i \in \mathbb{N}_{n+1}^* : \forall t \in W_i^{(\omega)} : && C_{i,t,2}^{(\omega)} \leftarrow Q_{i,t,2}^{(\omega)}; \\ &\forall i \in \mathbb{N}_{n+1}^* : \forall t \in \mathbb{N}_{n_i} \setminus W_i^{(\omega)} : && C_{i,t,2}^{(\omega)} \xleftarrow{\$} \mathbb{G}_1. \end{aligned}$$

and publishes the ciphertext:

$$\{\tilde{C}^{(\omega)}, C_0^{(\omega)}, C_{0,1}^{(\omega)}, \{C_{0,0,2}^{(\omega)}, C_{0,\varrho,2}^{(\omega)}\}, \{C_{i,1}^{(\omega)}, \{\underline{C_{i,t,2}^{(\omega)}}\}_{t=0}^{n_i-1}\}_{i=1}^n\}.$$

Note that $\mathcal{S}(\mathcal{D}_{\phi\omega})$ never needs to publish the value $Q_{j,j',2}^{(\omega)}$ it couldn't compute before.

$\mathcal{S}(\mathcal{D}_{\phi\omega})$ generates all records $\psi \neq \omega$ in the same way (including the re-randomization of all ciphertext components $Q_{i,t,2}^{(\psi)}$ not in the policy), except that it does not embed any DDH challenge in them.

The key issuing phase may be repeated.

$\mathcal{D}_{\phi\omega}$ may now perform queries for the record ψ and key φ ($\mathcal{S}(\mathcal{D}_{\phi\omega})$ extracts φ and ψ from ZKPoK_4). If the key satisfies the ciphertext policy of the queried record, P' is computed honestly. Else the key doesn't satisfy the policy: if $(\varphi, \psi) < (\phi, \omega)$, P' is computed randomly ; else if $(\varphi, \psi) \geq (\phi, \omega)$, P' is computed as in Game 7:

$$P' \leftarrow \left(\frac{\hat{C}^{(\psi)} \prod_{i=0}^n \mathrm{e}(C_{i,1}^{(\psi)}, D_{i,1}^{(\varphi)})}{M^{(0)} \mathrm{e}(C_0^{(\psi)}, D_0^{(\varphi)}) \prod_{i=1}^n \mathrm{e}(Q_{i,L_i^{(\varphi)},2}^{(\psi)}, D_{i,2}^{(\varphi)})} \right)^{k_c k_d} . \tag{13}$$

There are however two non-trivial cases of $\mathrm{e}(Q_{j,L_j^{(\varphi)},2}^{(\psi)}, D_{j,2}^{(\varphi)}) \stackrel{\text{def}}{=} \wp_j^{(\psi,\varphi)}$:

- To compute this pairing for an honest key $\varphi \neq \phi$, but for the record $\psi = \omega$ containing the DDH challenge: since $\lambda_j^{(\varphi)}$ (the discrete logarithm of $D_{j,2}^{(\varphi)}$) is known to $\mathcal{S}(\mathcal{D}_{\phi\omega})$, the pairing can be computed thus: $\wp_j^{(\omega,\varphi)} \leftarrow \mathrm{e}(g_1, g_2^\delta)^{\lambda_j^{(\varphi)}}$.
- To compute this pairing for the key $\varphi = \phi$ containing the DDH challenge and the record $\psi = \omega$ also containing the DDH challenge, the pairing can be computed thus: $\wp_j^{(\omega,\phi)} = \mathrm{e}(g_1, g_2^z)$. If z is random, then $\wp_j^{(\omega,\phi)}$ is computed according to Game 7-ϕ-ω, and if $z = \gamma\delta$, then $\wp_j^{(\omega,\phi)}$ is computed according to the lexicographically preceding game.

Finally $\mathcal{D}_{\phi\omega}$ outputs a guess b of which game it is in. $\mathcal{S}(\mathcal{D}_{\phi\omega})$ returns the same guess b for the DDH challenge. For there to be any difference between the current game and the preceding one, the key ϕ must not satisfy the policy of the record ω. This means at least one attribute in the key is not in the policy of the record; the probability that $\mathcal{S}(\mathcal{D}_{\phi\omega})$ hits such an attribute is therefore at least $N_{\text{attributes}}^{-1}$, and in all other cases it aborts and takes a random guess.

$$\mathbf{Adv}_{\mathcal{D}}^{\text{Game 7–8}} \leq \max_{\phi,\omega} \left(N_{\text{rec.}} N_{\text{users}} \cdot \mathbf{Adv}_{\mathcal{D}_{\phi\omega}}^{\text{Game 7-}\phi\text{-}\omega} \right) \leq \max_{\omega,\phi} \left(N_{\text{attr.}} N_{\text{rec.}} N_{us.} \cdot \mathbf{Adv}_{\mathcal{S}(\mathcal{D}_{\phi\omega})}^{\text{SXDH}} \right) = \mathit{negl.}$$

5.7 Proof of Lemma 9

If there exists a PPT algorithm $\mathcal{D}$ which distinguishes between Game 8 and Game 9 with non-negligible advantage, then we can construct a PPT algorithm $\mathcal{S}(\mathcal{D})$ which has rewindable black-box access to $\mathcal{D}$ and which plays the security game of HABE with non-negligible advantage.

Again, we consider a series of hybrid games Game 8-0 to Game 8-N_{records}, where in Game 8-ω every record $\psi \leq \omega$ is handled as in Game 9, and all records $\psi > \omega$ are handled as in Game 8. Clearly, Game 8-0 is exactly Game 8 and Game 8-N_{records} is exactly Game 9.

The existence of $\mathcal{D}$ implies the existence of $\mathcal{D}_\omega$ (for some ω) that can distinguish between Game 8-$(\omega - 1)$ and Game 8-ω with advantage N_{records}^{-1} that of $\mathcal{D}$.

The challenger computes the issuer public key and sends it to $\mathcal{S}(\mathcal{D}_\omega)$. $\mathcal{S}(\mathcal{D}_\omega)$ relays it to $\mathcal{D}_\omega$.

$\mathcal{S}(\mathcal{D}_\omega)$ runs $\mathcal{D}_\omega$, who may now requests a key with attribute list $\{L_i\}$ where $L_0 = 0$. $\mathcal{S}(\mathcal{D}_\omega)$ relays the request to the challenger and the response back to $\mathcal{D}_\omega$. This can be repeated polynomially many times.

When $\mathcal{D}_\omega$ wants to encrypt a record ψ:

- If $\psi < \omega$, $\mathcal{S}(\mathcal{D}_\omega)$ encrypts a random plaintext $M^{(\psi)\prime} \xleftarrow{\$} \mathbb{G}_{\mathrm{T}}$ under a random policy $W^{(\psi)\prime}$ (but with $W_0^{(\psi)\prime} = \{0\}$). Again, $C_{0,0,2}^{(\psi)}$ is remembered but not published.
- For the challenge record $\psi = \omega$, $\mathcal{S}(\mathcal{D}_\omega)$ sets $M^{(0)} = M^{(\omega)}$ and $W^{(0)} = W^{(\omega)}$, as well as $M^{(1)} \xleftarrow{\$} \mathbb{G}_{\mathrm{T}}$ and selects $W^{(1)}$ randomly (but with $W_0^{(1)} = \{\varrho\}$), and sends these to the HABE challenger. The challenger flips a bit $b \xleftarrow{\$} \mathbb{N}_2$ and encrypts $M^{(b)}$ under policy $W^{(b)}$ (we shall denote the resulting ciphertext $C^{(b)}$). $\mathcal{S}(\mathcal{D}_\omega)$ relays the ciphertext to $\mathcal{D}_\omega$.
- If $\psi > \omega$, $\mathcal{S}(\mathcal{D}_\omega)$ encrypts the plaintext $M^{(\psi)}$ under policy $W'^{(\psi)} = \{\{0, \varrho\}, W_1^{(\psi)}, W_2^{(\psi)}, \ldots, W_n^{(\psi)}\}$ (including the "issuer" attribute). $\mathcal{S}(\mathcal{D}_\omega)$ remembers the ciphertext component $C_{0,0,2}^{(\psi)}$ corresponding to the "issuer" attribute but doesn't publish it. $\mathcal{S}(\mathcal{D}_\omega)$ sends the rest of the ciphertext to $\mathcal{D}_\omega$.

$\mathcal{S}(\mathcal{D}_\omega)$ runs $\mathcal{D}_\omega$, who may now ask for more keys (polynomially many times).

$\mathcal{S}(\mathcal{D}_\omega)$ runs $\mathcal{D}_\omega$, who may perform a query with a key φ on record ψ ($\mathcal{S}(\mathcal{D}_\omega)$ extracts φ and ψ from ZKPoK_6). $\mathcal{S}(\mathcal{D}_\omega)$ checks if the key φ satisfies the ciphertext policy $W^{(\psi)}$. If the key doesn't satisfy the policy, then $\mathcal{S}(\mathcal{D}_\omega)$ computes P' randomly. P' is thus distributed exactly as in both games. If it does then:

- For records $\psi < \omega$, P' is computed as follows:

$$P' \leftarrow \left(\frac{\hat{C}^{(\psi)} \prod_{i=0}^{n} \mathrm{e}(C_{i,1}^{(\psi)}, D_{i,1}^{(\varphi)})}{M^{(\psi)} \, \mathrm{e}(C_0^{(\psi)}, D_0^{(\varphi)}) \prod_{i=1}^{n} \mathrm{e}(C_{i,L_i^{(\varphi)},2}^{(\psi)}, D_{i,2}^{(\varphi)})} \right)^{k_c k_d} . \tag{14}$$

- For records $\psi = \omega$, P' is computed as follows:

$$P' \leftarrow \left(\frac{\hat{C}^{(b)} \prod_{i=0}^{n} \mathrm{e}(C_{i,1}^{(b)}, D_{i,1}^{(\varphi)})}{M^{(0)}\, \mathrm{e}(C_0^{(b)}, D_0^{(\varphi)}) \prod_{i=1}^{n} \mathrm{e}(C_{i,L_i^{(\varphi)},2}^{(b)}, D_{i,2}^{(\varphi)})} \right)^{k_c k_d}. \tag{15}$$

  If $b = 0$ then $P' = \mathrm{e}(C_{0,\varrho,2}^{(\omega)}, D_{0,2}^{(\varphi)})^{k_\varrho^{-1} k_c k_d}$ and thus distributed exactly as in Game 8-($\omega - 1$). If $b = 1$ then P' is distributed exactly as in Game 8-ω.

- For records $\psi > \omega$, P' is computed as:

$$P' \leftarrow \mathrm{e}(C_{0,0,2}^{(\psi)}, D_{0,2}^{(\varphi)})^{k_c k_d}.$$

$\mathcal{S}(\mathcal{D}_\omega)$ sends the result P' back to $\mathcal{D}_\omega$.

The query phase may be repeated polynomially many times.

$\mathcal{S}(\mathcal{D}_\omega)$ runs $\mathcal{D}_\omega$, who now finally outputs a guess of b. $\mathcal{S}(\mathcal{D}_\omega)$ sends b to the challenger. $\mathcal{S}(\mathcal{D}_\omega)$ has the same advantage in the HABE security game, as $\mathcal{D}_\omega$ has in distinguishing between Game 8-($\omega - 1$) and Game 8-ω.

$$\mathbf{Adv}_{\mathcal{D}}^{\text{Game 8–9}} \leq \max_{\omega} \left(N_{\text{records}} \cdot \mathbf{Adv}_{\mathcal{D}_\omega}^{\text{Game 8-}(\omega\,-\,1)\text{–8-}\omega} \right) \leq \max_{\omega} \left(N_{\text{records}} \cdot \mathbf{Adv}_{\mathcal{S}(\mathcal{D}_\omega)}^{\text{HABE}} \right) = \mathit{negl}.$$

5.8 Proof of Lemma 10

If there exists a PPT algorithm $\mathcal{D}$ which distinguishes between Game 6 and Game 7 with non-negligible advantage, we can construct a PPT algorithm $\mathcal{S}(\mathcal{D})$ (which has rewindable black box access to $\mathcal{D}$) which can break the semantic security of El-Gamal (which would in turn break the SXDH assumption) with advantage $(N_{\text{users}} N_{\text{records}} N_{\text{categories}})^{-1}$ that of $\mathcal{D}$, which is still a non-negligible advantage.

$\mathcal{S}(\mathcal{D})$ receives an El-Gamal public key $\bar{X} = g_2^{\bar{x}}$ from the challenger of the El-Gamal semantic-security game. $\mathcal{S}(\mathcal{D})$ chooses $\bar{m}_0, \bar{m}_1 \xleftarrow{\$} \mathbb{Z}_p^*$. It computes $\bar{M}_0 = g_2^{\bar{m}_0}$ and $\bar{M}_1 = g_2^{\bar{m}_1}$ and gives them to the challenger. The challenger flips a coin $\bar{b}$ and gives $\mathcal{S}(\mathcal{D})$ the ciphertexts $\bar{C}_1 = \bar{M}_{\bar{b}} X^{\bar{r}}$, $\bar{C}_2 = g_2^{\bar{r}}$. $\mathcal{S}(\mathcal{D})$ proceeds as follows: it chooses a key ϕ at random from the set of honest users and a value $\bar{\jmath}$ at random. In the key issuing protocol for key ϕ it sends $\bar{X}$ as its El-Gamal public key. It sends $\bar{C}_1$ and $\bar{C}_2$ instead of $E_{3,\bar{\jmath}}^{(\phi)}$ and $F_{3,\bar{\jmath}}^{(\phi)}$. All other values of $E_{3,i}$ and $F_{3,i}$ can be computed normally (these are simply El-Gamal encryptions). $\mathcal{S}(\mathcal{D})$ then fakes ZKPoK_4 using the rewindable black-box access to $\mathcal{D}$.

It extracts the values λ_i' and $a_{i,t}$ from $\mathcal{A}$ in ZKPoK_5. $\mathcal{S}(\mathcal{D})$ can therefore recover the values of $\lambda_i^{(\phi)} = \lambda_i' + \lambda_i''$ for all $i \neq \bar{\jmath}$ and, using the escrow functionality, can compute $g_1^{\epsilon_{i,t}}$ from the ciphertext components $C_{i,t,2}$.

With non-negligible probability, there will be a query in which a key φ that does not satisfy the ciphertext policy of a given record, can decrypt that record anyway (if that is not the case then $\mathcal{D}$ cannot have a non-negligible advantage). If $\varphi \neq \phi$, $\mathcal{S}(\mathcal{D})$ aborts and takes a random guess. If the ciphertext policy of record satisfies $L_{\bar{\jmath}}^{(\phi)} \in W_{\bar{\jmath}}$, then $\mathcal{S}(\mathcal{D})$ aborts. In all other cases we have $\epsilon_{\bar{\jmath}, L_{\bar{\jmath}}^{(\phi)}} \neq 0 \pmod{p}$ and $0 = \sum_{i=0}^{n} \epsilon_{i, L_i^{(\varphi)}} \lambda_i^{(\varphi)} \pmod{p}$.

By setting $\lambda_{\bar{\jmath}}$ to $m_b + \lambda_{\bar{\jmath}}'$, $\mathcal{S}(\mathcal{D})$ can test if:

$$g_1^0 \stackrel{?}{=} \prod_{i=0}^{n} \left(g_1^{\epsilon_{i,L_i}} \right)^{\lambda_i}.$$

$\mathcal{S}(\mathcal{D})$ outputs the value of b for which the above equation is satisfied.

$$\mathbf{Adv}_{\mathcal{D}}^{\text{Game 6–7}} \leq N_{\text{records}} N_{\text{users}} N_{\text{categories}} \cdot \mathbf{Adv}_{\mathcal{S}(\mathcal{D}_\omega)}^{\text{El-Gamal}} = \mathit{negl}.$$

6 Alternative Construction

We now sketch the main idea of how one could instantiate our scheme based on the first construction of Nishide et al. [NYO08]. They proved their scheme secure under static assumptions (DBDH and D-Lin) and not directly in the generic bilinear group model. The price to pay is that the alternative HABE scheme is not (match-concealing) secure, but only *selectively* secure; the security proof of our alternative construction is therefore somewhat awkward. Our alternative scheme is secure under the D-Lin, DBDH and SXDH assumptions, as well as the $\max(N_{\text{users}}, N_{\text{records}})$-SFP assumption in both $\mathbb{G}_1$ and $\mathbb{G}_2$.

This alternative construction shares some similarity with our main construction. The major difference is that each attribute has two public key components. The "missing piece" is now composed of two parts which must not be revealed separately: the query protocol is substantially modified, and the privacy of the user is now computational (as opposed to perfect in our main construction).

Readers are advised to read Appendix A or [NYO08, Section 3.4] before continuing reading this section.

6.1 Selective Security

In the selective security game of HABE, which we formally define in Appendix A.2.1, the adversary of the HABE game must announce which two policies we wishes to be challenged on before he receives the system public key. $\mathcal{E}$ must provide a (polynomially-sized) set of ciphertext policies that are going to be used by honest databases.

This means that in the security proof we need to guess the policy of the record the adversary attacks, *before we publish the issuer public key.* We can achieve this by requiring either:

- that the total number of attributes in the scheme is logarithmic in the security parameter, or
- that $\mathcal{E}$ publishes a polynomial-sized set of policies before the start of the protocol, and that all honest databases only use ciphertext policies from that set.

Let N_{policies} be the number of policies that can be used by honest databases. We assume that this number is polynomial w.r.t. the security parameter.

6.2 The Construction

6.2.1 Group and CRS Setup

These are identical to our main construction.

6.2.2 Issuer Setup and Check Issuer Key

We generate the issuer public and private key according to the equations of the HABE scheme (see Appendix A.2.2), again with the special zeroth category containing only one "Issuer" attribute. Finally, a signing key pair that is good for message vectors of length 2 (instead of 1) in $\mathbb{G}_2$ is generated.

Checking the issuer key proceeds along the same lines as in our main construction.

6.2.3 Database Setup and Check Database Key

During database setup, the public and private key components corresponding to the "Database ϱ" attribute are computed as follows:

$$k_{a\varrho}, k_{b\varrho} \xleftarrow{\$} \mathbb{Z}_p, \qquad A_{0,\varrho}^{a_{0,\varrho}} \leftarrow \left(A_{0,0}^{a_{0,0}}\right)^{k_{a\varrho}}, \qquad A_{0,\varrho}^{b_{0,\varrho}} \leftarrow \left(A_{0,0}^{b_{0,0}}\right)^{k_{b\varrho}}.$$

The database also generates a signing key pair that is good for message vectors of length 2 in $\mathbb{G}_1$.

Checking the database key proceeds along the same lines as in our main construction.

6.2.4 Key Generation

We can easily adapt the joint key computation in our main scheme to the key generation algorithm of the alternative HABE scheme (cf. Appendix A.2.3). The aim remains the same: make sure neither the issuer nor the user know the values λ_i for $i \in \mathbb{N}_{n+1}^*$ (the s_i can be determined by the issuer and don't need to be generated jointly).

Also, since there are now two key components per attribute, the issuer needs to compute the signature σ on the tuple $\{D_{0,1}, D_{0,2}\}$ (instead of just $D_{0,2}$).

6.2.5 Record Issuing

The database computes the ciphertext as in the alternative HABE scheme (see Appendix A.2.4), except that it does not publish the ciphertext components corresponding to the "Issuer" attribute. It also generates a signature σ on the tuple $\{C_{0,\varrho,1}, C_{0,\varrho,2}\}$. Finally it issues a non-interactive GS proof to certify it computed the ciphertext component "Database ϱ" correctly:

$$\mathsf{NIZK}_{3'}\left\{(r, -r_{0,\varrho}) : C_0 = g_1^{r} \wedge C_{0,\varrho,1}^{-r_{0,\varrho}} = (A_{0,\varrho}^{b_{0,\varrho}}) \wedge C_{0,\varrho,2} = (A_{0,\varrho}^{a_{0,\varrho}})^{r}(A_{0,\varrho}^{a_{0,\varrho}})^{-r_{0,\varrho}}\right\}.$$

6.2.6 Decrypting a Record

The oblivious transfer protocol must be changed substantially, since we never want the user to see the two parts of the "missing piece" $P_1' = \mathrm{e}(D_{0,1}^{k_{d1}}, C_{0,\varrho,1}^{k_{c1}})^{k_{b\varrho}^{-1}}$ and $P_2' = \mathrm{e}(D_{0,2}^{k_{d2}}, C_{0,\varrho,2}^{k_{c2}})^{k_{a\varrho}^{-1}}$ separately (we are unable to prove the scheme secure otherwise). We will make use of the homomorphic property of the El-Gamal encryption to achieve this. The protocol remains 10-move.

The ZKPoK_8 in Figure 4 and Equation 16 is used to verify that the user knows the signature to $\{C_{0,\varrho,1}^{(\psi)}, C_{0,\varrho,2}^{(\psi)}\}$ and $\{D_{0,1}^{(\varphi)}, D_{0,2}^{(\varphi)}\}$ (first four lines), that both El-Gamal encryptions are correct (next 2 lines), and that k_{c1d1}^{-1} and k_{c2d2}^{-1} are the products $k_{c1}^{-1} k_{d1}^{-1}$ and $k_{c2}^{-1} k_{d2}^{-1}$ respectively (last line). Both parties parse vk_ϱ as $\{g_Z', f_Z', g_R', f_U', g_M', g_N', f_M', f_N', A', B'\}$ and vk_I as $\{g_Z'', f_Z'', g_R'', f_U'', g_M'', g_N'', f_M'', f_N'', A'', B''\}$.

$$
\begin{array}{lcl}
\underline{\mathsf{U}^{(\varphi)}\left(CT^{(\psi)}, SU^{(\varphi)}, PD^{(\varrho)}, PI\right):} & \text{Anonymous channel} & \underline{\mathsf{DB}^{(\varrho)}(SD^{(\varrho)}, PI):} \\
k_{c1}, k_{c2}, k_{d1}, k_{d2}, r_6, r_7, x \xleftarrow{\$} \mathbb{Z}_p^*, & & \\
C'_1 \leftarrow (C^{(\psi)}_{0,\varrho,1})^{k_{c1}}, \quad C'_2 \leftarrow (C^{(\psi)}_{0,\varrho,2})^{k_{c2}}, & & \\
D''_1 \leftarrow (D^{(\varphi)}_{0,1})^{k_{d1}}, \quad D''_2 \leftarrow (D^{(\varphi)}_{0,2})^{k_{d2}}, & & \\
\sigma' \leftarrow \mathsf{SigRerand}(\sigma^{(\psi)}), & & \\
\sigma'' \leftarrow \mathsf{SigRerand}(\sigma^{(\varphi)}), & & \\
X \leftarrow g_{\mathrm{T}}^{x},\ S_1 \leftarrow \mathfrak{U}_{1,2}^{k_{d1}},\ S_2 \leftarrow \mathfrak{U}_{2,1}^{k_{d2}}, & \xrightarrow{\quad X, S_1, S_2 \quad} & \\
E_6 \leftarrow \mathrm{e}(C^{(\psi)}_{0,\varrho,1}, D^{(\varphi)}_{0,1})X^{r_6},\ F_6 \leftarrow g_{\mathrm{T}}^{r_6}, & & \\
E_7 \leftarrow \mathrm{e}(C^{(\psi)}_{0,\varrho,2}, D^{(\varphi)}_{0,2})X^{r_7},\ F_7 \leftarrow g_{\mathrm{T}}^{r_7}. & \xrightarrow{\quad E_6, F_6, E_7, F_7 \quad} & r_8 \xleftarrow{\$} \mathbb{Z}_p^*, \\
\text{Parse } \sigma' \text{ and } \sigma'' \text{ as respecively} & & \\
\{Z', R', S', T', U', V', W'\} \text{ and} & \xrightarrow{\quad C'_1, C'_2, S', T', V', W' \quad} & E_8 \leftarrow E_6^{k_{b\varrho}^{-1}} E_7^{k_{a\varrho}^{-1}} X^{r_8}, \\
\{Z'', R'', S'', T'', U'', V'', W''\}. & \xrightarrow{\quad D''_1, D''_2, S'', T'', V'', W'' \quad} & F_8 \leftarrow F_6^{k_{b\varrho}^{-1}} F_7^{k_{a\varrho}^{-1}} g_{\mathrm{T}}^{r_8}. \\
 & \xrightarrow{\quad \mathsf{ZKPoK}_8 \text{ (see Equation 16)} \quad} & \\
 & \xleftarrow{\quad E_8, F_8 \quad} & \\
P \leftarrow E_8 F_8^{-x}. & \xleftarrow{\quad \mathsf{ZKPoK}_9 \text{ (see Equation 17)} \quad} & \\
\text{Compute } M' \text{ from Equation 18.} & & \\
\text{If } M' \text{ correct: } \textbf{return } M'; & & \textbf{return } \varepsilon \\
\text{else } \textbf{return } \perp . & &
\end{array}
$$

Figure 4: Transfer protocol of alternative construction.

$$
\begin{aligned}
\mathsf{ZKPoK}_8\Big\{ & \Big(\boxed{k_{c1}^{-1}}, \boxed{k_{c2}^{-1}}, \boxed{Z'}, \boxed{R'}, \boxed{U'}, \boxed{k_{d1}^{-1}}, \boxed{k_{d2}^{-1}}, \boxed{Z''}, \boxed{R''}, \boxed{U''}, \boxed{x}, \boxed{r_7}, \boxed{r_8}, \boxed{k_{c1d1}^{-1}}, \boxed{k_{c2d2}^{-1}}\Big) : \\
& A' = \mathrm{e}(\boxed{Z'}, g'_Z)\,\mathrm{e}(\boxed{R'}, g'_R)\,\mathrm{e}(T', S')\,\mathrm{e}(C'_1, g'_M)^{\boxed{k_{c1}^{-1}}}\,\mathrm{e}(C'_2, g'_N)^{\boxed{k_{c2}^{-1}}} \wedge \\
& B' = \mathrm{e}(\boxed{Z'}, f'_Z)\,\mathrm{e}(\boxed{U'}, f'_U)\,\mathrm{e}(W', V')\,\mathrm{e}(C'_1, f'_M)^{\boxed{k_{c1}^{-1}}}\,\mathrm{e}(C'_2, f'_N)^{\boxed{k_{c2}^{-1}}} \wedge \\
& A'' = \mathrm{e}(g''_Z, \boxed{Z''})\,\mathrm{e}(g''_R, \boxed{R''})\,\mathrm{e}(S'', T'')\,\mathrm{e}(g''_M, D''_1)^{\boxed{k_{d1}^{-1}}}\,\mathrm{e}(g''_N, D''_2)^{\boxed{k_{d2}^{-1}}} \wedge \\
& B'' = \mathrm{e}(f''_Z, \boxed{Z''})\,\mathrm{e}(f''_U, \boxed{U''})\,\mathrm{e}(V'', W'')\,\mathrm{e}(f''_M, D''_1)^{\boxed{k_{d1}^{-1}}}\,\mathrm{e}(f''_N, D''_2)^{\boxed{k_{d2}^{-1}}} \wedge \\
& E_6 = \mathrm{e}(C'_1, D''_1)^{\boxed{k_{c1d1}^{-1}}} X^{\boxed{r_6}} \wedge F_6 = g_{\mathrm{T}}^{\boxed{r_6}} \wedge \\
& E_7 = \mathrm{e}(C'_2, D''_2)^{\boxed{k_{c2d2}^{-1}}} X^{\boxed{r_7}} \wedge F_7 = g_{\mathrm{T}}^{\boxed{r_7}} \wedge X = g_{\mathrm{T}}^{\boxed{x}} \wedge \\
& \mathfrak{U}_{1,2} = S_1^{\boxed{k_{d1}^{-1}}} \wedge \mathfrak{U}_{1,2}^{\boxed{k_{c1}^{-1}}} = S_1^{\boxed{k_{c1d1}^{-1}}} \wedge \mathfrak{U}_{2,1} = S_2^{\boxed{k_{d2}^{-1}}} \wedge \mathfrak{U}_{2,1}^{\boxed{k_{c2}^{-1}}} = S_1^{\boxed{k_{c2d2}^{-1}}} \Big\}.
\end{aligned}
\tag{16}
$$

$$
\begin{aligned}
\mathsf{ZKPoK}_9\Big\{ & \Big(\boxed{k_{a\varrho}^{-1}}, \boxed{k_{b\varrho}^{-1}}, \boxed{r_8}\Big) : \Big(A_{0,\varrho}^{a_{0,\varrho}}\Big)^{\boxed{k_{a\varrho}^{-1}}} = A_{0,0}^{a_{0,0}} \wedge \Big(A_{0,\varrho}^{b_{0,\varrho}}\Big)^{\boxed{k_{b\varrho}^{-1}}} = A_{0,0}^{b_{0,0}} \wedge \\
& E_8 = E_6^{\boxed{k_{b\varrho}^{-1}}} E_7^{\boxed{k_{a\varrho}^{-1}}} X^{\boxed{r_8}} \wedge F_8 = F_6^{\boxed{k_{b\varrho}^{-1}}} F_7^{\boxed{k_{a\varrho}^{-1}}} g_{\mathrm{T}}^{\boxed{r_8}} \Big\}.
\end{aligned}
\tag{17}
$$

$$M' \leftarrow \frac{\hat{C}P \prod_{i=1}^{n} \mathrm{e}(C_{i,L_i,1}, D_{i,1})\, \mathrm{e}(C_{i,L_i,2}, D_{i,2})}{\mathrm{e}(C_0, D_0) \prod_{i=0}^{n} \mathrm{e}(C_0, D_{i,0})}. \tag{18}$$

If the user key satisfies the policy, then $M = M'$.

6.3 Security Proof of Alternative Construction

Completeness follows from construction. It is also easy to adapt the escrow functionality of the issuer.

6.3.1 Corrupted Database or Corrupted Issuer+Database

The security proof is very similar.

A user can decrypt a record iff: $\sum_{i=0}^{n} a_{i,L_i} \lambda_i^{(\varphi)} \epsilon_{i,L_i}^{(\psi)} = 0$. All values of $a_{i,j}$ are non-zero, as the public key is rejected if it contains neutral group elements. If only the database is malicious, then it never sees the random values λ_i. The probability of a user decrypting even if his key doesn't satisfy the policy of the record is therefore less than p^{-1}. If the issuer and database are malicious, we can reduce this case to the El-Gamal semantic security game as we did in the proof of our main contribution.

DDH in groups $\mathbb{G}_{\mathrm{T}}$ and $\mathbb{G}_2$ guarantees that the user has (computational) privacy in the transfer protocol (we first replace E_6 by a random number (DDH in $\mathbb{G}_{\mathrm{T}}$ (El-Gamal)), then we replace E_7 by a random number (idem), then S_1 (DDH in $\mathbb{G}_2$—we embed the challenge in the CRS $\mathfrak{U}$), then S_2 (idem)).

6.3.2 Corrupted User

Games 1 to 6 from our main contribution can easily be adapted to the alternative construction. We will not restate the modified games, which we denote by Games 1' to 6', here.

Game 7' $\mathcal{S}_{7'}$ runs like $\mathcal{S}_{6'}$ except that instead of computing E_8 and F_8 honestly in the transfer protocol it computes:

$$E_8 \leftarrow PX^{r_8}, \qquad F_8 \leftarrow g_{\mathrm{T}}^{r_8}.$$

Where the "missing piece" P is computed honestly (using x):

$$P \leftarrow \mathrm{e}(C_{0,\varrho,1}^{(\psi)}, D_{0,1}^{(\varphi)})^{k_{b\varrho}}\, \mathrm{e}(C_{0,\varrho,2}^{(\psi)}, D_{0,2}^{(\varphi)})^{k_{a\varrho}}.$$

Using the rewindable black-box access of $\mathcal{A}$, $\mathcal{S}_{7'}$ fakes ZKPoK_9. The difference between Games 6' and 7' is zero, since all values have exactly the same distribution in both games.

Game 8' Is similar to Game 7. P is computed thus:

$$P \leftarrow \frac{M\, \mathrm{e}(C_0, D_0) \prod_{i=0}^{n} \mathrm{e}(C_0, D_{i,0})}{\hat{C} \prod_{i=1}^{n} \mathrm{e}(C_{i,L_i,1}, D_{i,1})\, \mathrm{e}(Q_{i,L_i,2}, D_{i,2})}.$$

$$\text{Where } \forall i \in \mathbb{N}_{n+1}^{*} \forall t \in \mathbb{N}_{n_i} : Q_{i,t,2}^{(\psi)} \begin{cases} \leftarrow C_{i,t,2}^{(\psi)} & \text{if } t \in W_i^{(\psi)}; \\ \xleftarrow{\$} \mathbb{G}_1 & \text{if } t \notin W_i^{(\psi)}. \end{cases}$$

The difference between Game 7' and Game 8' is negligible. We omit the proof, since it is very similar to the proof in Section 5.5, except that $\mathcal{S}(\mathcal{D}_\omega)$ needs to guess (before the start of the game) the policy that will be used for the challenge record, and abort if his guess was wrong. The advantage loss will be an additional factor N_{policies}^{-1}.

Game 9' Is similar to Game 8. If the key φ does not satisfy the policy of record ψ, $\mathcal{S}_{9'}$ computes $P \xleftarrow{\$} \mathbb{G}_T$. The difference between the two games is negligible. The proof is very similar to the proof in Section 5.6 and is omitted.

Game 10' Is similar to Game 9. $\mathcal{S}_{10'}$ also encrypts a random message under a random policy. If the key φ doesn't satisfy the policy of record ψ, P is computed randomly. If the key does satisfy:

$$P \leftarrow \frac{M^{(\psi)}\,\mathrm{e}(C_0, D_0)\prod_{i=0}^{n}\mathrm{e}(C_0, D_{i,0})}{\hat{C}\prod_{i=1}^{n}\mathrm{e}(C_{i,L_i,1}, D_{i,1})\,\mathrm{e}(C_{i,L_i,2}, D_{i,2})}.$$

P is such that the user recovers the "real" plaintext $M^{(\psi)}$ at the close of the transfer protocol.

The difference between the two games is negligible. We omit the proof, since it is very similar to the proof in Section 5.7, except that $\mathcal{S}(\mathcal{D}_\omega)$ needs to guess (before the start of the game) the policy that will be used for the challenge record, and abort if his guess was wrong. The advantage loss will be an additional factor N_{policies}^{-1}.

7 Implementation

The source code of the implementation is IBM-confidential and will not be published as part of this thesis.

We implemented the scheme presented in Section 4 in the C++ programming language and using the PBC Library[23] [Lyn07]. We wrote our own library to handle the interactive and non-interactive zero-knowledge proofs.[24] Our implementation currently works on "D"-type curves [Lyn07], which, according to the author of the library, are type-3 curves where the SDXH assumption is believed to hold. We also wrote a workaround in our code for a bug we found in the PBC library.[25]

We wrote a separate program for each of the algorithms described in Section 2.3,[26] and one program for each of the parties for each of the protocols.[27] All programs have a command-line interface. We wrote a script (`preparedemo.sh`), which runs a semi-automatic demonstration of the scheme as a whole: it opens separate terminals for each of the following parties: the trusted CRS generator, the issuer, the database, and the user; and gives instruction for running the programs.

All keys and ciphertexts are stored in files on disk (in the demo, the issuer, database, and user each have a separate folder) in an ad-hoc data format. Group elements from $\mathbb{G}_1$ and $\mathbb{G}_2$ are represented with the standard elliptic-curve point compression technique[28] to reduce storage and transmission size. All programs communicate via TCP/IP, but no effort is made to establish secure channels. All programs also abort when they detect an error.

When a value supposedly in $\mathbb{G}_1$, $\mathbb{G}_2$, or $\mathbb{G}_\mathrm{T}$ is read from disk for the first time, or read from the network, the library checks that it is on the elliptic curve, and our code checks that its order is either 1 or p.

[23]The PBC library can be downloaded at `http://crypto.stanford.edu/pbc/`.

[24]We used the six-move variant of the perfect ZKPoK, instead of the four-move variant.

[25]In version 0.5.10, when generating a random element from $\mathbb{G}_2$, the result may not lie in a cyclic group of order p, but in a group of order a multiple of p. According to the author of the library this does not cause problems in general, "because it turns out the pairing computation works the same for all coset representatives". However, our code checks the order of all elements received from untrusted sources, and therefore fails. See `http://groups.google.com/group/pbc-devel/browse_thread/thread/3af23f1c261a8b80`.

[26]IssuerSetup, DatabaseSetup, IssueRecord, CheckRecord, as well as the CRS generator.

[27]CheckIssuerKey for $\mathcal{I}$, $\mathcal{D}$, and $\mathcal{U}$; IssueKey for $\mathcal{I}$, and $\mathcal{U}$; CheckDatabaseKey for $\mathcal{D}$, and $\mathcal{U}$; Query for $\mathcal{D}$ and $\mathcal{U}$.

[28]Representing only the x-coordinate of the point and the sign of the y-coordinate

Our program can be used to encrypt files of arbitrary size instead of just elements of $\mathbb{G}_T$: We derive a symmetric key from the plaintext group element $M \in \mathbb{G}_T$ and then encrypt the file with an authenticated encryption scheme, specifically AES-256-GCM (which is a NIST standard).[29] If a user does not have permissions to access a record, he will derive an incorrect symmetric key from the recovered plaintext group element M', and thus the decryption of the symmetric ciphertext will fail with an error message.

We have not fully optimized the scheme: the implementation can be made faster by changing the way some of the computation is performed, for example by doing pre-computations, multi-exponentiations, and multi-pairings. We believe that more careful programming can lower the run-time of some parts of the code by a factor of 1.5–2 (especially in the ZKPoK protocols).

7.1 Measured Efficiency

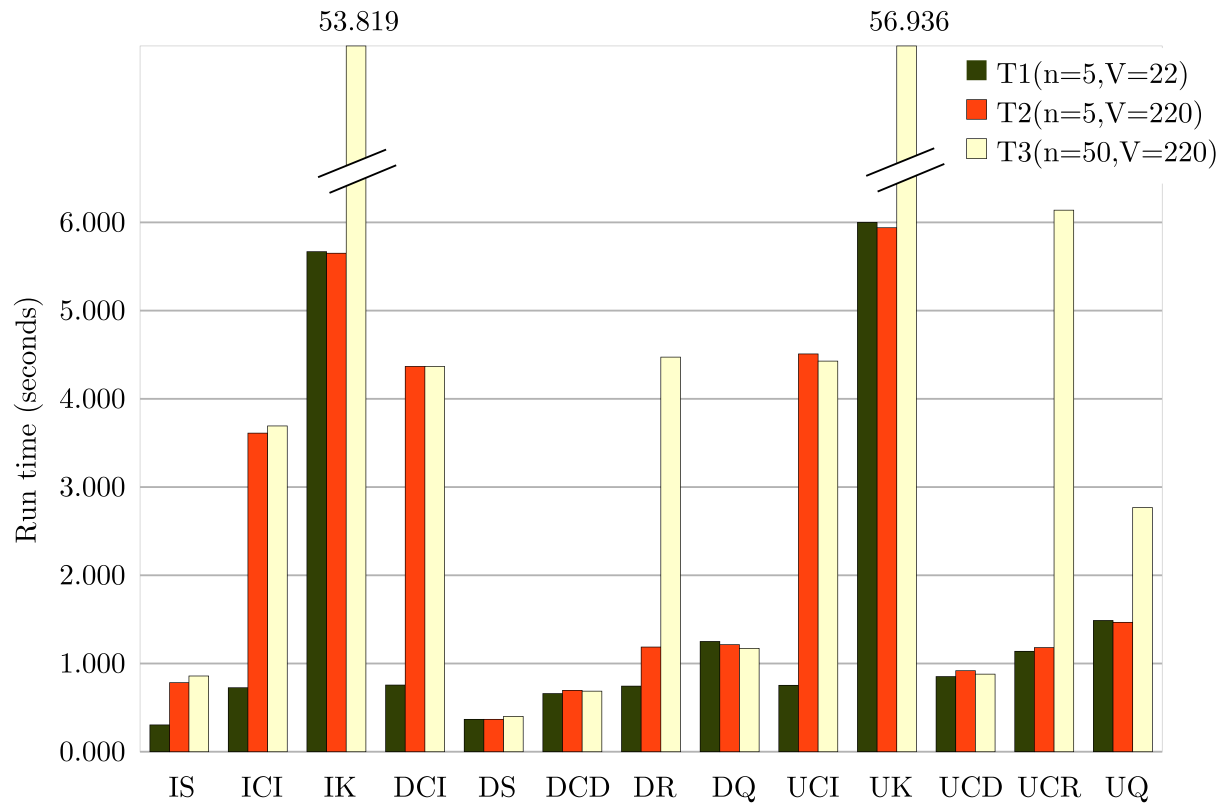

Figure 5: Run times (excluding idle time) of the algorithms and protocols in our implementation. T1 uses the same policies and attributes as in the example of Section 2.2 and as shown in Figure 1: using $\approx 2^{158}$ group size, 5 categories, and 22 attributes total. Between T1 and T2 we multiply the number of attributes by 10 while keeping the number of categories constant. Between T2 and T3 we multiply the number of categories by 10 while keeping the number of attributes constant. We recommend you contrast these results with the theoretical predictions of Table 3.
IS is IssuerSetup, ICI is CheckIssuerKey for $\mathcal{I}$, IK is IssueKey for $\mathcal{I}$. DCI is CheckIssuerKey for $\mathcal{D}$, DS is DatabaseSetup, DCD is CheckDatabaseKey for $\mathcal{D}$, DR is IssueRecord, DQ is Query for $\mathcal{D}$. UCI is CheckIssuerKey for $\mathcal{U}$, UK is IssueKey for $\mathcal{U}$, UCD is CheckDatabaseKey for $\mathcal{U}$, UCR is CheckRecord, UQ is Query for $\mathcal{U}$.

[29]We use the Crypto++ library for that. It's available at `http://www.cryptopp.com/`.

We performed measurements of the run times of all the programs of our implementation. The results are summarized in Figure 5. The experiments were run on Ubuntu Linux 10.10, x86 (32 bit) architecture, on an Intel T2600 processor clocked at 2.16GHz. All TCP/IP communication ran on localhost. The reported run times *exclude* idle time: we summed the “user” and “sys” components of the `time` utility (and ignored the “real” component); for algorithms the error introduced by this choice are too small to be noticeable; for protocols, the times of both parties should be added together to get a good estimate of the “real” run time.

We performed 4 experiments:

- **T1**: We used the example of Section 2.2 (and Figure 1) with a group size of $\approx 2^{158}$.[30] There are 5 categories and 22 attributes total in this experiment. The policy of the record had 10 well-formed ciphertext components, and 12 random ciphertext components (same as Figure 1).
- **T2**: Same $\approx 2^{158}$ group order, 5 categories, 220 attributes total. The policy of the record contained 100 well-formed ciphertext components.
- **T3**: Same $\approx 2^{158}$ group order, 50 categories, 220 attributes total. The policy of the record contained 100 well-formed ciphertext components.
- **T4**: The setting of T2, but with a $\approx 2^{289}$ group order.[31]

A graph of the run times of experiments T1–T3 are shown in Figure 5 (the exact figures are available with the source of this report). The run times under setting T4 (larger group elements) were between 2.90 and 3.62 times slower than the T2 setting, suggesting that the security parameter κ contributes to the overall run time as $O(\kappa^2)$ instead of the expected $O(\kappa^3)$.

The run times of our benchmark closely reflect the theoretical predictions of Table 3.[32]

We observe that most run times are low enough to be reasonable for practical applications.

8 Extensions

8.1 Revocation of Keys

Key revocation lists (KRLs) allows the issuer to revoke certain keys he issued without compromising the privacy guarantees of the users. The KRLs are public, and users can prove to the database in zero-knowledge that their key has not been revoked. The concept we are going to describe now has been introduced by Nakanishi et al. [NFHF09] and used in a recent paper by Camenisch et al. [CDNZ11].

8.1.1 Additional Security Guarantees

Database Security Cheating users who have been revoked cannot successfully engage in a query protocol with an honest database (assuming the honest database has received a copy of the latest KRL).

[30]Using the `d159.param` curve (D type curve with discriminant 62003, $\approx 2^{159}$ base field size, $\approx 2^{158}$ group order).

[31]Using a D type curve with discriminant 285707, $\approx 2^{311}$ base field size, $\approx 2^{289}$ group order. This curve was found with the PBC library’s `listmnt` program, and generated with the library’s `gendparam` program.

[32]The algorithms/protocols IS, ICI, DCI and UCI are supposed to run in time linear to the number of attributes; IK, UK, UCR and UQ in time linear to the number of categories; DR in time linear to the sum of the number of categories and attributes; DS, DCD, DQ, UCD in constant-time.

User Security If a user engages successfully in a query protocol, the only information the database can deduce is that the user's key has not been revoked. The identity, and choice of record of that user remain private.

8.1.2 Modified and Additional Algorithms

To make KRL deployment possible, the interfaces of the IssuerSetup algorithm and Query protocol of Section 2.3 must be modified, and new algorithms GenerateKRL and CheckKRL are introduced.

- IssuerSetup takes an additional parameter N_{krlsize}, an estimate of the total number of users in the system.
- GenerateKRL: $\mathcal{I}(PK, SK, F) \overset{\$}{\rightarrow} KRL^{(krlid)}$.
  The issuer runs this algorithm to revoke all users in the subset F. The issuer needs to provide his public and secret keys. The output of the algorithm is a key revocation list $KRL^{(krlid)}$, where *krlid* is counter that is incremented each time this function is called.
- CheckKRL: $\forall \mathcal{R} \in \mathcal{U} \cup \mathcal{D}$: $\mathcal{R}(KRL^{(krlid)}) \overset{\$}{\rightarrow} b$.
  Upon receiving a new KRL from the issuer, each user and database runs this algorithm to test whether it is correctly formed or not. The output is a bit b indicating the result of this check.
- In the Query protocol, the user takes the most recent KRL $KRL^{(krlid)}$ as additional input, and the database takes the serial number *krlid* of the most recent KRL as additional input. The protocol fails if the user and database have a different version of the KRL, or if the user is on the list of revoked users in the KRL. Internally, the user performs a zero-knowledge set-membership proof [CCS08] for the database that he is in the set of non-revoked users.

8.1.3 Modified Real and Ideal World

The environment may send a new message $\langle \mathcal{E}, \mathcal{I}, \text{“UpdateKRL”}, F\rangle$ at any time after issuer setup, where F is a subset of revoked users, to the issuer. In the real world $\mathcal{I}$ calls GenerateKRL and broadcasts the new KRL to all users and databases ($\forall \mathcal{R} \in \mathcal{U} \cup \mathcal{D}$). The latter then run CheckKRL, and if the output is 1, they relay the subset F to the environment $\langle \mathcal{R}, \mathcal{E}, \text{“NewKRL”}, F\rangle$. In the ideal world, $\mathcal{I}$ relays the message $\langle \mathcal{I}, \mathsf{T}, \text{“UpdateKRL”}, F\rangle$ to T. T saves the subset F and broadcasts the message $\langle \mathsf{T}, \forall \mathcal{R} \in \mathcal{U} \cup \mathcal{D}, \text{“NewKRL”}, F\rangle$ to all users and databases, who in turn relay $\langle \mathcal{R}, \mathsf{T}, \text{“NewKRL”}, F\rangle$ to the environment.

When in the ideal world, T receives a “Query” message from user $\mathcal{U}_\varphi$, it additionally checks if U_φ is in the set of revoked users F; if so, T aborts the query protocol.

8.1.4 Construction

We need to modify the IssueKey and Query protocols, modify the IssuerSetup and CheckIssuerKey algorithms, and describe the GenerateKRL and CheckKRL algorithms.

IssuerSetup The signing and verification key pair sgk_I and vk_I must now be able to sign a message consisting of two elements of $\mathbb{G}_2$ (instead of just one).

The issuer generates a new signing and verification key pair sgk_K and vk_K which are able to sign messages consisting of three elements of $\mathbb{G}_1$, and a new signing and verification key pair sgk_R and vk_R which are able to sign messages consisting of one element of $\mathbb{G}_1$.

The issuer then generates $N_{\text{krlsize}} + 1$ signatures: $\{\hat{\sigma}_i \leftarrow \mathsf{Sign}(g_1^i, sgk_R)\}_{i=0}^{N_{\text{krlsize}}}$.

The issuer adds the signing keys sgk_K and sgk_R to his private key. He adds the verification keys vk_K and vk_R, as well as all signatures $\{\hat{\sigma}_i\}_{i=0}^{N_{\text{krlsize}}}$ to his public key (notice the public key has now size linear in the number of users).

CheckIssuerSetup The issuer additionally needs to prove knowledge of his new signing keys. The users and databases need to check signatures $\{\hat{\sigma}_i\}_{i=0}^{N_{\text{krlsize}}}$ with vk_R.

IssueKey The issuer issues the signature σ not on $D_{0,2}$, but instead on $\{D_{0,2}, g_2^{\varphi}\}$ (φ, the user index, now becomes explicit).

GenerateKRL The issuer generates a set of intervals of valid keys $\{start_i, end_i\}_{i=1}^{N_{\text{intervals}}}$ (where $N_{\text{intevals}} \in \mathbb{N}$, $start_i, end_i \in \mathbb{Z}_p$), where the union of all intervals $\bigcup_{i=1}^{N_{\text{intervals}}}[start_i, end_i]$ contains all non-revoked users IDs, and contains no revoked user ID. Furthermore, we require than $0 \le end_i - start_i \le N_{\text{krlsize}}$.

The issuer increments the counter *krlid* and generates signatures on the following message tuple with his key sgk_K:

$$\{\varsigma_i \leftarrow \mathsf{Sign}(\{g_1^{krlid}, g_1^{start_i}, g_1^{end_i}\}, sgk_K)\}_{i=1}^{N_{\text{intervals}}}. \tag{19}$$

The KRL consists of the list of intervals and their signature: $\{krlid, \{start_i, end_i, \varsigma_i\}_{i=1}^{N_{\text{intervals}}}\}$.

CheckKRL Upon receiving a KRL, the user and database need to check that all the signatures ς_i are correct, and that $\forall i \in \mathbb{N}^*_{\text{intervals}+1} : (end_i - start_i) \in \mathbb{N}_{N_{\text{krlsize}}+1}$.

Query In the query phase, the user needs to find the interval j in the most recent KRL $KRL^{(krlid)}$ containing his user ID φ: $start_j \le \varphi \le end_j$. (The user can find such an interval if and only if his key has not been revoked.) The user then proves possession of the following signatures in zero-knowledge to the database:

- The signature ς_j of $\{g_1^{krlid}, g_1^{\boxed{start_j}}, g_1^{\boxed{end_j}}\}$ under the key vk_K, where the user does not reveal $\boxed{start_j}$ and $\boxed{end_j}$, but reveals *krlid*.
- The signature $\hat{\sigma}_{end_j - \varphi}$ of $g_1^{\boxed{end_j} - \boxed{\varphi}}$ under the key vk_R, thus proving that $\boxed{\varphi} < \boxed{end_j}$.[33]
- The signature $\hat{\sigma}_{\varphi - start_j}$ of $g_1^{\boxed{\varphi} - \boxed{start_j}}$ under the key vk_R, thus proving that $\boxed{\varphi} > \boxed{start_j}$.
- The signature $\sigma^{(\varphi)}$ of $\{\boxed{D_{0,2}^{(\varphi)}}, g_2^{\boxed{\varphi}}\}$ under the key vk_I (similar to our main construction).
- The signature $\sigma^{(\psi)}$ of $\boxed{C_{0,\varrho,2}^{(\psi)}}$ under the key vk_ϱ (same as in our main construction).

8.1.5 Possible Improvements

There are many possible improvements for this membership proof, see for example Nakanishi et al. [NFHF09] or the paper derived from Rafik Chaabouni's master thesis [CCS08].

[33] Actually, φ and end_j are elements of $\mathbb{Z}_p$ but here you should consider them as integers in $\mathbb{N}_p$, so that the less-than operator is well defined.

8.2 Record Revocation Lists (RRL)

In the scheme as described, it is not possible for a database to revoke a record once it is issued. We can fix this problem by using the same idea as for revoking user keys, except that each database maintains its own RRL instead of the issuer. Each database must also publish its own signatures on the values $0, \ldots, N_{\text{krlsize}}$.

After a record has been revoked, only a dishonest issuer (through the escrow functionality), or a combination of dishonest database and dishonest user can recover the plaintext.

8.3 Key / Record Expiration

Key and record expiration can be handled with revocation lists. Each time a record or key expires, it is revoked by the issuer resp. the database.

If the key indexes and record indexes are ordered strictly by expiration date, the number of intervals inside the RRLs and KRLs will never increase, however the revocation lists have to be recomputed each time a key or a record expires.

If it is expected that a significant number of records will be revoked, it might become very expensive to re-generate the RRL/KRL each time. An alternative solution would be to add an additional group element representing the expiration time of the key/record in the signatures of the key or ciphertext components. The users would then have to do a range proof [CCS08] that the expiration time of their key and record are larger than the current time.

8.4 Anonymity Revocation

Under certain circumstances it might be desirable to revoke the user's anonymity of certain database queries. To that effect, we introduce a new player trusted by all parties: the anonymity revoker (AR). In practice AR could for example be a judge in a court of law. The AR publishes two El-Gamal public keys (one in $\mathbb{G}_1$ and one in $\mathbb{G}_2$). During each query phase, the user encrypts his unblinded value $D_{0,2}^{(\varphi)}$ and the unblinded ciphertext component $C_{0,\varrho,2}^{(\psi)}$ with the public keys of AR. The user then integrates the resulting El-Gamal ciphertext in ZKPoK_6.

The anonymity of the transaction can be revoked by asking the AR to decrypt the ciphertext. Note that the AR might choose to decrypt only the user identifier or only the record identifier.

8.5 Preventing Denial-of-service

We do not address Denial-of-Service issues in this paper. One could imagine that the users have to pay for the privilege of interacting with the database [CDN10] or do a proof-of-work similar to HashCash [Bac02].

8.6 Only Allow Databases Vetted by the Issuer

This is easy to implement: the database needs to get a signature of the issuer on it's value $A_{0,\varrho}$ (after a proof of knowledge that it knows k_ϱ) and publishes it with its public key. Honest users must check this signature before interacting with the database.

8.7 Allow Everybody to Publish Records

Instead of requiring that everybody who wishes to publish records has to setup a database, one could also imagine allowing users to issue records themselves, and "entrust" them to the database of their choice.

A user who wishes to publish a record, encrypts the record as described in Section 4.2.7 under a database's public key (except for the signature on $C_{0,\varrho,2}$), and computes NIZK_3 by himself. The user then proves in interactive zero-knowledge to the database that he knows the plaintext[34] and the policy of the ciphertext he wishes to entrust to the database. If the database accepts the proofs, it computes the signature on $C_{0,\varrho,2}$ and broadcasts the now complete ciphertext to all users.

If the database is corrupt, it can't decrypt the user's entrusted record without help from either the corrupted issuer, or a corrupted user who happens to have a key that satisfies the policy.

9 Conclusion

We created a scheme that allows a database to publish records that are protected by an hidden access control policy, and which users can access without revealing their identity or choice of record. An extension to our scheme allows the key issuer to revoke the user's keys. We have proved our scheme secure in the generic bilinear group model.

Our alternative construction removes the need for the full-fledged generic bilinear group model, at the cost however of introducing the unnatural concept of selective security.

Our construction uses attribute-based encryption, and is more efficient and allows for more powerful access control policies than prior work, which was based on anonymous credentials. Finally, we have implemented our scheme, and experimentally validated the theoretical run-time predictions.

9.1 Future Work

In the future it would be interesting to find a construction that would fix the major shortcomings of our scheme, namely: remove the unfettered access the issuer has over the published records, and prove our scheme secure under more standard and non-interactive assumptions than the generic bilinear group model.

A very recent paper by Okamoto and Takashima introduced the concept of fully secure functional encryption over prime order groups [OT10]. It would be interesting to see if we can adapt your results to work with their scheme.

[34] Instead of proving the clause $\hat{C} = \boxed{M}Y^{\boxed{r}}$, which we can't do with the sigma-protocol–based zero-knowledge proofs, the user proves the clause $\hat{C} = g_{\mathrm{T}}^{\boxed{m}}Y^{\boxed{r}}$, where $\boxed{m}$ is the discrete logarithm of $\boxed{M}$. In practice $\boxed{M}$ is computed randomly and used only to derive a symmetric key, so there is no problem in computing $\boxed{m} \xleftarrow{\$} \mathbb{Z}_p^*, \boxed{M} \leftarrow g_{\mathrm{T}}^{\boxed{m}}$.

References

[AFG$^+$10] Masayuki Abe, Georg Fuchsbauer, Jens Groth, Kristiyan Haralambiev, and Miyako Ohkubo. Structure-preserving signatures and commitments to group elements. In Tal Rabin, editor, *Advances in Cryptology CRYPTO 2010*, volume 6223 of *Lecture Notes in Computer Science*, pages 209–236. Springer Berlin / Heidelberg, 2010. (Cited on pages 7, 16, 20, 21 and 63.)

[AHO10] Masayuki Abe, Kristiyan Haralambiev, and Miyako Ohkubo. Signing on elements in bilinear groups for modular protocol design. Cryptology ePrint Archive, Report 2010/133, 2010. `http://eprint.iacr.org/`. (Cited on pages 7, 16, 17, 21 and 63.)

[ASM06] Man Au, Willy Susilo, and Yi Mu. Constant-size dynamic k-taa. In Roberto De Prisco and Moti Yung, editors, *Security and Cryptography for Networks*, volume 4116 of *Lecture Notes in Computer Science*, pages 111–125. Springer Berlin / Heidelberg, 2006. (Cited on page 20.)

[Bac02] Adam Back. Hashcash – A denial of service counter-measure. 2002. `http://www.cypherspace.org/hashcash/hashcash.pdf`. (Cited on page 48.)

[BB08] Dan Boneh and Xavier Boyen. Short signatures without random oracles and the sdh assumption in bilinear groups. *Journal of Cryptology*, 21:149–177, 2008. (Cited on pages 17 and 20.)

[BFM88] Manuel Blum, Paul Feldman, and Silvio Micali. Non-interactive zero-knowledge and its applications. In *Proceedings of the twentieth annual ACM symposium on Theory of computing*, STOC '88, pages 103–112, New York, NY, USA, 1988. ACM. (Cited on page 18.)

[BSW07] John Bethencourt, Amit Sahai, and Brent Waters. Ciphertext-policy attribute-based encryption. In *Proceedings of the 2007 IEEE Symposium on Security and Privacy*, SP '07, pages 321–334, Washington, DC, USA, 2007. IEEE Computer Society. (Cited on page 7.)

[Can00] Ran Canetti. Universally composable security: A new paradigm for cryptographic protocols. Cryptology ePrint Archive, Report 2000/067, 2000. (Cited on page 11.)

[CCS08] Jan Camenisch, Rafik Chaabouni, and Abhi Shelat. Efficient protocols for set membership and range proofs. In Josef Pieprzyk, editor, *Advances in Cryptology - ASIACRYPT 2008*, volume 5350 of *Lecture Notes in Computer Science*, pages 234–252. Springer Berlin / Heidelberg, 2008. (Cited on pages 46, 47 and 48.)

[CDM00] Ronald Cramer, Ivan Damgrd, and Philip MacKenzie. Efficient zero-knowledge proofs of knowledge without intractability assumptions. In Hideki Imai and Yuliang Zheng, editors, *Public Key Cryptography*, volume 1751 of *Lecture Notes in Computer Science*, pages 354–373. Springer Berlin / Heidelberg, 2000. (Cited on pages 7, 17, 24, 56, 58, 59 and 60.)

[CDN09] Jan Camenisch, Maria Dubovitskaya, and Gregory Neven. Oblivious transfer with access control. In *Proceedings of the 16th ACM conference on Computer and communications security*, CCS '09, pages 131–140, New York, NY, USA, 2009. ACM. (Cited on pages 8 and 11.)

[CDN10] Jan Camenisch, Maria Dubovitskaya, and Gregory Neven. Unlinkable priced oblivious transfer with rechargeable wallets. In Radu Sion, editor, *Financial Cryptography and Data Security*, volume 6052 of *Lecture Notes in Computer Science*, pages 66–81. Springer Berlin / Heidelberg, 2010. (Cited on page 48.)

[CDNZ11] Jan Camenisch, Maria Dubovitskaya, Gregory Neven, and Gregory Zaverucha. Oblivious transfer with hidden access control policies. In Dario Catalano, Nelly Fazio, Rosario Gennaro, and Antonio Nicolosi, editors, *Public Key Cryptography PKC 2011*, volume 6571 of *Lecture Notes in Computer Science*, pages 192–209. Springer Berlin / Heidelberg, 2011. (Cited on pages 6, 8, 11, 15, 18, 26 and 45.)

[CL04] Jan Camenisch and Anna Lysyanskaya. Signature schemes and anonymous credentials from bilinear maps. In Matt Franklin, editor, *Advances in Cryptology CRYPTO 2004*, volume 3152 of *Lecture Notes in Computer Science*, pages 56–72. Springer Berlin / Heidelberg, 2004. (Cited on page 20.)

[CM09] Sanjit Chatterjee and Alfred Menezes. On Cryptographic Protocols Employing Asymmetric Pairings – The Role of Ψ Revisited. Cryptology ePrint Archive, Report 2009/480, 2009. `http://eprint.iacr.org/`. (Cited on page 16.)

[CNS07] Jan Camenisch, Gregory Neven, and Abhi Shelat. Simulatable adaptive oblivious transfer. In Moni Naor, editor, *Advances in Cryptology - EUROCRYPT 2007*, volume 4515 of *Lecture Notes in Computer Science*, pages 573–590. Springer Berlin / Heidelberg, 2007. (Cited on pages 6, 7, 8, 9, 11 and 56.)

[CS97a] J. Camenisch and M. Stadler. Proof systems for general statements about discrete logarithms. *Institute for Theoretical Computer Science, ETH Zrich, Tech. Rep. TR*, 260, 1997. (Cited on pages 56 and 59.)

[CS97b] Jan Camenisch and Markus Stadler. Efficient group signature schemes for large groups. In Burton Kaliski, editor, *Advances in Cryptology CRYPTO '97*, volume 1294 of *Lecture Notes in Computer Science*, pages 410–424. Springer Berlin / Heidelberg, 1997. 10.1007/BFb0052252. (Cited on page 17.)

[GO94] Oded Goldreich and Yair Oren. Definitions and properties of zero-knowledge proof systems. *Journal of Cryptology*, 7:1–32, 1994. (Cited on page 18.)

[GPS08] Steven D. Galbraith, Kenneth G. Paterson, and Nigel P. Smart. Pairings for cryptographers. *Discrete Applied Mathematics*, 156(16):3113 – 3121, 2008. Applications of Algebra to Cryptography. (Cited on pages 7 and 16.)

[GS08] Jens Groth and Amit Sahai. Efficient non-interactive proof systems for bilinear groups. In Nigel Smart, editor, *Advances in Cryptology EUROCRYPT 2008*, volume 4965 of *Lecture Notes in Computer Science*, pages 415–432. Springer Berlin / Heidelberg, 2008. (Cited on pages 7, 18, 29, 30, 32, 61, 62 and 63.)

[GSW10] Essam Ghadafi, Nigel. Smart, and Bogdan Warinschi. Grothsahai proofs revisited. In Phong Nguyen and David Pointcheval, editors, *Public Key Cryptography PKC 2010*, volume 6056 of *Lecture Notes in Computer Science*, pages 177–192. Springer Berlin / Heidelberg, 2010. (Cited on pages 18 and 63.)

[KM06] Neal Koblitz and Alfred Menezes. Another look at generic groups. In *Advances in Mathematics of Communications*, pages 13–28, 2006. (Cited on page 17.)

[KSW08] Jonathan Katz, Amit Sahai, and Brent Waters. Predicate encryption supporting disjunctions, polynomial equations, and inner products. In Nigel Smart, editor, *Advances in Cryptology EUROCRYPT 2008*, volume 4965 of *Lecture Notes in Computer Science*, pages 146–162. Springer Berlin / Heidelberg, 2008. (Cited on page 7.)

[LOS$^+$10] Allison Lewko, Tatsuaki Okamoto, Amit Sahai, Katsuyuki Takashima, and Brent Waters. Fully secure functional encryption: Attribute-based encryption and (hierarchical) inner product encryption. In Henri Gilbert, editor, *Advances in Cryptology EUROCRYPT 2010*, volume 6110 of *Lecture Notes in Computer Science*, pages 62–91. Springer Berlin / Heidelberg, 2010. (Cited on pages 7 and 8.)

[Lyn07] Lynn, Ben. *On the Implementation of Pairing-Based Cryptography.* PhD thesis, Stanford University, June 2007. (Cited on page 43.)

[NFHF09] Toru Nakanishi, Hiroki Fujii, Yuta Hira, and Nobuo Funabiki. Revocable group signature schemes with constant costs for signing and verifying. In Stanislaw Jarecki and Gene Tsudik, editors, *Public Key Cryptography PKC 2009*, volume 5443 of *Lecture Notes in Computer Science*, pages 463–480. Springer Berlin / Heidelberg, 2009. (Cited on pages 6, 8, 45 and 47.)

[Nis08] Takashi Nishide. *Cryptographic Schemes with Minimum Disclosure of Private Information in Attribute-Based Encryption and Multiparty Computation.* PhD thesis, University of Electro-Communications, September 2008. (Cited on pages 6, 8, 19 and 54.)

[NP99] Moni Naor and Benny Pinkas. Oblivious transfer with adaptive queries. In Michael Wiener, editor, *Advances in Cryptology CRYPTO 99*, volume 1666 of *Lecture Notes in Computer Science*, pages 573–590. Springer Berlin / Heidelberg, 1999. (Cited on page 8.)

[NYO08] Takashi Nishide, Kazuki Yoneyama, and Kazuo Ohta. Attribute-based encryption with partially hidden encryptor-specified access structures. In Steven Bellovin, Rosario Gennaro, Angelos Keromytis, and Moti Yung, editors, *Applied Cryptography and Network Security*, volume 5037 of *Lecture Notes in Computer Science*, pages 111–129. Springer Berlin / Heidelberg, 2008. (Cited on pages 6, 7, 8, 9, 16, 18, 19, 20, 24, 39, 54 and 55.)

[OT10] Tatsuaki Okamoto and Katsuyuki Takashima. Fully secure functional encryption with general relations from the decisional linear assumption. In Tal Rabin, editor, *Advances in Cryptology CRYPTO 2010*, volume 6223 of *Lecture Notes in Computer Science*, pages 191–208. Springer Berlin / Heidelberg, 2010. (Cited on page 49.)

[PW00] Birgit Pfitzmann and Michael Waidner. Composition and integrity preservation of secure reactive systems. In *Proceedings of the 7th ACM conference on Computer and communications security*, CCS '00, pages 245–254, New York, NY, USA, 2000. ACM. (Cited on page 11.)

[PW01] B. Pfitzmann and M. Waidner. A model for asynchronous reactive systems and its application to secure message transmission. In *Security and Privacy, 2001. S&P 2001. Proceedings. 2001 IEEE Symposium on*, pages 184–200. IEEE, 2001. (Cited on page 11.)

[Sho97] Victor Shoup. Lower bounds for discrete logarithms and related problems. In *Proceedings of the 16th annual international conference on Theory and application* *of cryptographic techniques*, EUROCRYPT'97, pages 256–266, Berlin, Heidelberg, 1997. Springer-Verlag. (Cited on page 17.)

[Sho04] Victor Shoup. Sequences of games: a tool for taming complexity in security proofs. Cryptology ePrint Archive, Report 2004/332, 2004. `http://eprint.iacr.org/`. (Cited on pages 28 and 30.)

[SW05] Amit Sahai and Brent Waters. Fuzzy identity-based encryption. In Ronald Cramer, editor, *Advances in Cryptology EUROCRYPT 2005*, volume 3494 of *Lecture Notes in Computer Science*, pages 457–473. Springer Berlin / Heidelberg, 2005. (Cited on page 7.)

[Yao82] Andrew C. Yao. Protocols for secure computations. *Foundations of Computer Science, Annual IEEE Symposium on*, 0:160–164, 1982. (Cited on page 8.)

A Hidden–ciphertext-policy Attribute-based Encryption

In this section we present the full construction of both Hidden–ciphertext-policy Attribute-based Encryption schemes by Nishide et al. [NYO08], so that the interested reader may compare it with our construction.

A.1 Match-concealing Secure Construction

The second construction of Nishide et al.'s HABE scheme [NYO08]—which is the one we use in our main contribution—is proven secure in the generic bilinear group setting [Nis08], and requires type-3 pairings. It allows the issuer to add attributes and categories after system set-up. Unlike our scheme, the issuer is assumed to be trusted.

A.1.1 Trusted Issuer Setup

The trusted issuer chooses the number of categories n and the number of different attributes possible for each category $\{n_i\}_{i=1}^n$. He computes:

$$\left\{\{a_{i,t} \xleftarrow{\$} \mathbb{Z}_p^*\}_{t=1}^{n_i}\right\}_{i=1}^n, \qquad w, \beta \xleftarrow{\$} \mathbb{Z}_p^*,$$
$$\left\{\{A_{i,t} \leftarrow g_1^{a_{i,t}}\}_{t=1}^{n_i}\right\}_{i=1}^n, \qquad Y \leftarrow g_{\mathrm{T}}^w, \qquad B \leftarrow g_1^\beta.$$

The secret key of the issuer is $\{w, \beta, \{\{a_{i,t}\}_{t=1}^{n_i}\}_{i=1}^n\}$.
His public key is $\{Y, B, \{\{A_{i,t}\}_{t=1}^{n_i}\}_{i=1}^n, n, \{n_i\}_{i=1}^n\}$.

A.1.2 User Key Generation

Let $L = [L_1 \in \mathbb{N}_{n_1}, L_2 \in \mathbb{N}_{n_2}, \ldots, L_n \in \mathbb{N}_{n_n}]$ be the attribute list the user wishes to get in his key. The issuer computes:

$$s \xleftarrow{\$} \mathbb{Z}_p^*, \qquad \{\lambda_i \xleftarrow{\$} \mathbb{Z}_p^*\}_{i=1}^n,$$
$$D_0 \leftarrow g_2^{(w+s)/\beta}, \qquad \left\{D_{i,1} \leftarrow g_2^{s+a_{i,L_i}\lambda_i}\right\}_{i=1}^n, \qquad \left\{D_{i,2} \leftarrow g_2^{\lambda_i}\right\}_{i=1}^n.$$

The key of the user is $\left\{D_0, (L_i, D_{i,1}, D_{i,2})_{i=1}^n\right\}$.

A.1.3 Issue Record

Let $M \in \mathbb{G}_{\mathrm{T}}$ be the record to encrypt, and let $W^{(\psi)} = [W_0^{(\psi)}, \ldots, W_n^{(\psi)}]$ where $\forall i \in \mathbb{N}_{n+1}^* : W_i \subset \mathbb{N}_{n_i}$ be the hidden ciphertext policy.

The encryptor computes the following values:

$$\left\{r_i \xleftarrow{\$} \mathbb{Z}_p^*\right\}_{i=1}^n, \qquad r \leftarrow \sum_{i=1}^n r_i \pmod p, \qquad \hat{C} \leftarrow MY^r,$$
$$\left\{\{\epsilon_{i,t} \leftarrow 0\}_{\forall t \in W_i}\right\}_{i=1}^n, \qquad \left\{\{\epsilon_{i,t} \xleftarrow{\$} \mathbb{Z}_p^*\}_{\forall t \in \mathbb{N}_{n_i} \setminus W_i}\right\}_{i=1}^n, \qquad C_0 \leftarrow B^r,$$
$$\left\{C_{i,1} \leftarrow g_1^{r_i}\right\}_{i=1}^n, \qquad \left\{\left\{C_{i,t,2} \leftarrow A_{i,t}^{r_i} g_1^{\epsilon_{i,t}}\right\}_{t=1}^{n_i}\right\}_{i=1}^n.$$

The ciphertext is: $\left\{\hat{C}, C_0, \{C_{i,1}, \{C_{i,t,2}\}_{t=1}^{n_i}\}_{i=1}^n\right\}$.

A.1.4 Offline Record Decryption

Given a key with attribute list $L = [L_1, \dots L_n]$, the user computes the following value when he wishes to decrypt a record:

$$M' \leftarrow \frac{\hat{C} \prod_{i=1}^{n} \mathrm{e}(C_{i,1}, D_{i,1})}{\mathrm{e}(C_0, D_0) \prod_{i=1}^{n} \mathrm{e}(C_{i,L_i,2}, D_{i,2})} = M g_{\mathrm{T}}^{-\sum_{i=1}^{n} \lambda_i \epsilon_{i,L_i}}. \tag{20}$$

The recovered message M' is equal to M if the key satisfies the record policy (i.e., if all ϵ_{i,L_i} are equal to zero). Intuitively, if the key does not satisfy the policy of the record, then the message is blinded by a random value, and the user cannot decrypt.

A.2 Selectively Secure Construction

The first construction of Nishide et al.'s HABE scheme—which we use in our alternative construction—is proven *selectively* secure under the DBDH and D-Lin assumption [NYO08], and can be used with all 3 types of pairings. We will present the type-3 variant here, which we used in our alternate construction.

A.2.1 Selective security

In the selective security setting, the adversary $\mathcal{A}$ must commit to the challenge ciphertext policies $W^{(0)}$ and $W^{(1)}$ before he receives the system public key. The HABE Game in Section 3.5.1 is modified as follows [NYO08]:

(0) $\mathcal{A}$ sends the two ciphertext policies $W^{(0)}$ and $W^{(1)}$ to the challenger.

(1, 2) Same as in Section 3.5.1.

(3) $\mathcal{A}$ submits two messages $M^{(0)}$, $M^{(1)}$ to the challenger. The latter flips a coin b, encrypts $M^{(b)}$ under policy $W^{(b)}$ and sends the resulting ciphertext to $\mathcal{A}$.

(4, 5) Same as in Section 3.5.1.

A.2.2 Trusted Issuer Setup

The trusted issuer chooses the number of categories n and the number of different attributes possible for each category $\{n_i\}_{i=1}^{n}$. He computes:

$$\left\{\{a_{i,t}, b_{i,t}, c_{i,t} \xleftarrow{\$} \mathbb{Z}_p^*\}_{t=1}^{n_i}\right\}_{i=1}^{n}, \qquad w \xleftarrow{\$} \mathbb{Z}_p^*, \qquad \left\{\{A_{i,t} \leftarrow g_1^{c_{i,t}}\}_{t=1}^{n_i}\right\}_{i=1}^{n}, \qquad Y \leftarrow g_{\mathrm{T}}^{w}.$$

The secret key of the issuer is $\{w, \{\{a_{i,t}, b_{i,t}, c_{i,t}\}_{t=1}^{n_i}\}_{i=1}^{n}\}$.
His public key is $\{Y, \{\{A_{i,t}^{a_{i,t}}, A_{i,t}^{b_{i,t}}\}_{t=1}^{n_i}\}_{i=1}^{n}, n, \{n_i\}_{i=1}^{n}\}$.

A.2.3 User Key Generation

Let $L = [L_1 \in \mathbb{N}_{n_1}, L_2 \in \mathbb{N}_{n_2}, \dots, L_n \in \mathbb{N}_{n_n}]$ be the attribute list the user wishes to get in his key. The issuer computes:

$$\{s_i, \lambda_i \xleftarrow{\$} \mathbb{Z}_p^*\}_{i=1}^{n}, \qquad s \leftarrow \sum_{i=1}^{n} s_i \pmod{p}, \qquad D_0 \leftarrow g_2^{w-s},$$

$$\left\{D_{i,0} \leftarrow g_2^{s_i + a_{i,L_i} b_{i,L_i} c_{i,L_i} \lambda_i}\right\}_{i=1}^{n}, \qquad \left\{D_{i,1} \leftarrow g_2^{a_{i,L_i} \lambda_i}\right\}_{i=1}^{n}, \qquad \left\{D_{i,2} \leftarrow g_2^{b_{i,L_i} \lambda_i}\right\}_{i=1}^{n}.$$

The key of the user is $\big\{D_0, \{L_i, D_{i,0}, D_{i,1}, D_{i,2}\}_{i=1}^{n}\big\}$.

A.2.4 Issue Record

Let $M \in \mathbb{G}_{\mathrm{T}}$ be the record to encrypt, and let $W^{(\psi)} = [W_0^{(\psi)}, \dots, W_n^{(\psi)}]$ where $\forall i \in \mathbb{N}_{n+1}^{*} : W_i \subset \mathbb{N}_{n_i}$ be the hidden ciphertext policy.

The encryptor computes the following values:

$$\Big\{\Big\{r_{i,t} \xleftarrow{\$} \mathbb{Z}_p^{*}\Big\}_{t=1}^{n_i}\Big\}_{i=1}^{n}, \qquad r \xleftarrow{\$} \mathbb{Z}_p^{*}, \qquad \hat{C} \leftarrow MY^{r},$$

$$\Big\{\big\{\epsilon_{i,t} \leftarrow 0\big\}_{\forall t \in W_i}\Big\}_{i=1}^{n}, \qquad \Big\{\big\{\epsilon_{i,t} \xleftarrow{\$} \mathbb{Z}_p^{*}\big\}_{\forall t \in \mathbb{N}_{n_i} \setminus W_i}\Big\}_{i=1}^{n}, \qquad C_0 \leftarrow g_1^{r},$$

$$\Big\{\Big\{C_{i,t,1} \leftarrow A_{i,t}^{b_{i,t} r_{i,t}} g_1^{\epsilon_{i,t}}\Big\}_{t=1}^{n_i}\Big\}_{i=1}^{n}, \qquad \Big\{\Big\{C_{i,t,2} \leftarrow A_{i,t}^{a_{i,t}(r - r_{i,t})}\Big\}_{t=1}^{n_i}\Big\}_{i=1}^{n}.$$

The ciphertext is: $\Big\{\hat{C}, C_0, \big\{\{C_{i,t,1}, C_{i,t,2}\}_{t=1}^{n_i}\big\}_{i=1}^{n}\Big\}$.

A.2.5 Offline Record Decryption

Given a key with attribute list $L = [L_1, \dots, L_n]$, the user computes the following value when he wishes to decrypt a record:

$$M' \leftarrow \frac{\hat{C} \prod_{i=1}^{n} \mathrm{e}(C_{i,L_i,1}, D_{i,1})\, \mathrm{e}(C_{i,L_i,2}, D_{i,2})}{\mathrm{e}(C_0, D_0) \prod_{i=1}^{n} \mathrm{e}(C_0, D_{i,0})} = M g_{\mathrm{T}}^{\sum_{i=1}^{n} a_{i,L_i} \lambda_i \epsilon_{i,L_i}}.$$

The recovered message M' is equal to M if the key satisfies the record policy (i.e., if all ϵ_{i,L_i} are equal to zero). Intuitively, if the key does not satisfy the policy of the record, then the message is blinded by a random value, and the user cannot decrypt.

B Sigma-Protocol

Let $x \overset{\text{def}}{=} \big\{\big(\boxed{x_i} \in \mathbb{Z}_p\big)_{i=0}^{u_x}, \big(\boxed{\gamma_i} \in \mathbb{G}_1\big)_{i=0}^{u_g}, \big(\boxed{\eta_i} \in \mathbb{G}_2\big)_{i=0}^{u_h}\big\}$ and let $A : x \times y \mapsto \{0,1\}$ (1 meaning "satisfied" and 0 "not satisfied") be defined:

$$\bigwedge_{i=0}^{n_g}\Big(G_i = \prod_{j=0}^{n_{1,i}} g_{1,i,j}^{\boxed{x_{\nu_{1,i,j}}}}\Big) \bigwedge_{i=0}^{n_h}\Big(H_i = \prod_{j=0}^{n_{2,i}} h_{2,i,j}^{\boxed{x_{\nu_{2,i,j}}}}\Big) \bigwedge_{i=0}^{n_t}\Big(T_i = \prod_{j=0}^{n_{3,i}} t_{3,i,j}^{\boxed{x_{\nu_{3,i,j}}}} \prod_{j=0}^{n_{4,i}} \mathrm{e}(g_{4,i,j}, \boxed{\eta_{\nu_{4,i,j}}}) \prod_{j=0}^{n_{5,i}} \mathrm{e}(\boxed{\gamma_{\nu_{5,i,j}}}, h_{5,i,j})\Big),$$

where $y \overset{\text{def}}{=} \Big\{u_x, u_g, u_h, n_g, n_h, n_t, \big(n_{1,i}\big)_{i=0}^{n_g}, \big(n_{2,i}\big)_{i=0}^{n_h}, \big(n_{3,i}, n_{4,i}, n_{5,i}\big)_{i=0}^{n_t}, \in \mathbb{N},$
$\big(\big(\nu_{1,i,j}\big)_{j=0}^{n_{1,i}}\big)_{i=0}^{n_g}, \big(\big(\nu_{2,i,j}\big)_{j=0}^{n_{2,i}}\big)_{i=0}^{n_h}, \big(\big(\nu_{3,i,j}\big)_{j=0}^{n_{3,i}}\big)_{i=0}^{n_t} \in \mathbb{N}_{u_x+1},$
$\big(\big(\nu_{4,i,j}\big)_{j=0}^{n_{4,i}}\big)_{i=0}^{n_t} \in \mathbb{N}_{u_g+1},$
$\big(\big(\nu_{5,i,j}\big)_{j=0}^{n_{5,i}}\big)_{i=0}^{n_t} \in \mathbb{N}_{u_h+1},$
$\big(\big(t_{3,i,j}\big)_{j=0}^{n_{i,3}}, T_i\big)_{i=0}^{n_t} \in \mathbb{G}_{\mathrm{T}},$
$\big(\big(g_{1,i,j}\big)_{j=0}^{n_{i,1}}, G_i\big)_{i=0}^{n_g}, \big(\big(g_{4,i,j}\big)_{j=0}^{n_{i,4}}\big)_{i=0}^{n_t} \in \mathbb{G}_1,$
$\big(\big(h_{2,i,j}\big)_{j=0}^{n_{i,2}}, H_i\big)_{i=0}^{n_h}, \big(\big(h_{5,i,j}\big)_{j=0}^{n_{i,5}}\big)_{i=0}^{n_t} \in \mathbb{G}_2\Big\}.$

A Sigma protocol $\Sigma(\mathcal{P} \leftrightarrow \mathcal{V})$ is run between two PPT interactive machines $\mathcal{P}$ and $\mathcal{V}$. It is a protocol in which $\mathcal{P}$ can convince $\mathcal{V}$ that he knows a witness x that satisfies the relation $A(y, x)$ (on common input y), without revealing anything about x to the *honest* verifier [CS97a, CDM00, CNS07].

In Camenisch-Stadler notation (c.f. Equation 3 in Section 3.3), we write:

$$\Sigma\Big\{\Big((\boxed{x_i} \in \mathbb{Z}_p)_{i=0}^{u_x}, (\boxed{\gamma_i} \in \mathbb{G}_1)_{i=0}^{u_g}, (\boxed{\eta_i} \in \mathbb{G}_2)_{i=0}^{u_h}\Big) : \quad \bigwedge_{i=0}^{n_g}\Big(G_i = \prod_{j=0}^{n_{1,i}} g_{1,i,j}^{\boxed{x_{\nu_{1,i,j}}}}\Big)$$

$$\bigwedge_{i=0}^{n_h}\Big(H_i = \prod_{j=0}^{n_{2,i}} h_{2,i,j}^{\boxed{x_{\nu_{2,i,j}}}}\Big) \bigwedge_{i=0}^{n_t}\Big(T_i = \prod_{j=0}^{n_{3,i}} t_{3,i,j}^{\boxed{x_{\nu_{3,i,j}}}} \prod_{j=0}^{n_{4,i}} \mathrm{e}(g_{4,i,j}, \boxed{\eta_{\nu_{4,i,j}}}) \prod_{j=0}^{n_{5,i}} \mathrm{e}(\boxed{\gamma_{\nu_{5,i,j}}}, h_{5,i,j})\Big)\Big\}.$$

B.1 Properties of Sigma Protocols

3-move Let $¢_{\mathcal{P}}$ and $¢_{\mathcal{V}}$ be some random coins (a polynomial-length string of random bits),[35] let $A(y, x)$ be the relation associated with the sigma protocol, then $\Sigma(\mathcal{P} \leftrightarrow \mathcal{V})$ consists of the following deterministic polynomial-time algorithms:

- Commit : $\mathcal{P}(y, x, ¢_{\mathcal{P}}) \to a$.
- $\mathsf{Challenge}$: $\mathcal{V}(y, ¢_{\mathcal{V}}, a) \to c$, which *must* generate c *independently* of a.
- Open : $\mathcal{P}(y, x, ¢_{\mathcal{V}}, a, c) \to z$.
- Verify : $\mathcal{V}(y, ¢_{\mathcal{V}}, a, c, z) \to b$.

We also write $\mathrm{Out}_{\mathcal{V}}(\mathcal{P} \leftrightarrow \mathcal{V}) \overset{\text{def}}{=} b$.

Perfect completeness If $A(y, x) = 1$ then $\Pr_{¢_{\mathcal{V}}, ¢_{\mathcal{P}}}[\mathrm{Out}_{\mathcal{V}}(\mathcal{P} \leftrightarrow \mathcal{V}) \overset{\$?}{=} 1] = 1$.

Special soundness There exists a PPT algorithm $\mathcal{E}$, called the *extractor*, that, given access to two transcripts (a, c, z) and (a', c', z') where $a = a'$ and $c \neq c'$, can recover a witness x' such that $A(y, x') = 1$ with probability 1.

Special honest-verifier zero-knowledge There exists a PPT algorithm $\mathcal{S}$ (called the *simulator*) that on input c produces an accepting transcript (a, c, z). Furthermore, the transcripts produced by the simulator for random c's ($c \overset{\$}{\leftarrow} \mathbb{Z}_p; \mathcal{S}(c)$), and the transcripts produced by the honest prover and verifier $\mathcal{P} \leftrightarrow \mathcal{V}$ are perfectly indistinguishable.

B.2 Construction

In the following we will abuse the symbol $\overset{\$}{\leftarrow}$ for deterministic polynomial-time algorithms. It must be understood that these deterministic polynomial-time algorithms use their supply of random coins to generate random values.

B.2.1 Commit

The prover $\mathcal{P}$ computes:

$$(r_i \overset{\$}{\leftarrow} \mathbb{Z}_p)_{i=0}^{u_x}, \quad \Big(a_{g,i} \leftarrow \prod_{j=0}^{n_{1,i}} g_{1,i,j}^{r_{\nu_{1,i,j}}}\Big)_{i=0}^{n_g}, \quad \Big(a_{h,i} \leftarrow \prod_{j=0}^{n_{2,i}} h_{2,i,j}^{r_{\nu_{2,i,j}}}\Big)_{i=0}^{n_h}, \quad (\pi_i \overset{\$}{\leftarrow} \mathbb{G}_1)_{i=0}^{u_g},$$

$$(\rho_i \overset{\$}{\leftarrow} \mathbb{G}_2)_{i=0}^{u_h}, \quad \Big(a_{t,i} \leftarrow \prod_{j=0}^{n_{3,i}} t_{3,i,j}^{r_{\nu_{3,i,j}}} \prod_{j=0}^{n_{4,i}} \mathrm{e}(g_{4,i,j}, \rho_{\nu_{4,i,j}}) \prod_{j=0}^{n_{5,i}} \mathrm{e}(\pi_{\nu_{5,i,j}}, h_{5,i,j})\Big)_{i=0}^{n_t},$$

[35] This formalism allows us to transform all PPT interactive machines into deterministic machines with extra input.

and outputs the message $a \overset{\text{def}}{=} \left(\left(a_{g,i}\right)_{i=0}^{n_g}, \left(a_{h,i}\right)_{i=0}^{n_h}, \left(a_{t,i}\right)_{i=0}^{n_t}\right)$.

B.2.2 Challenge

The verifier $\mathcal{V}$ computes $c \overset{\$}{\leftarrow} \mathbb{Z}_p$ and outputs c.

B.2.3 Open

The prover $\mathcal{P}$ recomputes r_i, π_i, ρ_i from his random coins, computes:

$$\left(z_i \leftarrow r_i - c x_i \pmod p\right)_{i=0}^{u_x}, \qquad \left(\zeta_i \leftarrow \pi_i \gamma_i^{-c}\right)_{i=0}^{u_g}, \qquad \left(\xi_i \leftarrow \rho_i \eta_i^{-c}\right)_{i=0}^{u_h},$$

and outputs the message $z \overset{\text{def}}{=} \left(\left(z_i\right)_{i=0}^{n_g}, \left(\zeta_i\right)_{i=0}^{n_h}, \left(\xi_i\right)_{i=0}^{n_t}\right)$.

B.2.4 Verify

The verifier $\mathcal{V}$ outputs 1 if and only if the following holds:

$$\bigwedge_{i=0}^{n_g} \left(a_{g,i} \overset{?}{=} G_i^c \prod_{j=0}^{n_{1,i}} g_{1,i,j}^{z_{\nu_{1,i,j}}}\right) \bigwedge_{i=0}^{n_h} \left(a_{h,i} \overset{?}{=} H_i^c \prod_{j=0}^{n_{2,i}} h_{2,i,j}^{z_{\nu_{2,i,j}}}\right)$$
$$\bigwedge_{i=0}^{n_t} \left(a_{t,i} \overset{?}{=} T_i^c \prod_{j=0}^{n_{3,i}} t_{3,i,j}^{z_{\nu_{3,i,j}}} \prod_{j=0}^{n_{4,i}} \mathrm{e}(g_{4,i,j}, \xi_{\nu_{4,i,j}}) \prod_{j=0}^{n_{5,i}} \mathrm{e}(\zeta_{\nu_{5,i,j}}, h_{5,i,j})\right).$$

B.2.5 Extractor

From two transcripts (a, c, z) and (a', c', z') where $a = a'$ and $c \neq c'$ the extractor $\mathcal{E}$ can recover the witness as follows:

$$\left(x_i \leftarrow \frac{z_i - z_i'}{c' - c}\right)_{i=0}^{n_x}, \qquad \left(\gamma_i \leftarrow (\zeta_i/\zeta_i')^{1/(c'-c)}\right)_{i=0}^{n_g}, \qquad \left(\eta_i \leftarrow (\xi_i/\xi_i')^{1/(c'-c)}\right)_{i=0}^{n_h}.$$

B.2.6 Simulator

The simulator $\mathcal{S}$ receives an arbitrary c in his input, computes:

$$\left(z_i \overset{\$}{\leftarrow} \mathbb{Z}_p\right)_{i=0}^{u_x}, \quad \left(a_{g,i} \leftarrow G_i^c \prod_{j=0}^{n_{1,i}} g_{1,i,j}^{z_{\nu_{1,i,j}}}\right)_{i=0}^{n_g}, \quad \left(a_{h,i} \leftarrow H_i^c \prod_{j=0}^{n_{2,i}} h_{2,i,j}^{z_{\nu_{2,i,j}}}\right)_{i=0}^{n_h}, \quad \left(\zeta_i \overset{\$}{\leftarrow} \mathbb{G}_1\right)_{i=0}^{u_g},$$
$$\left(\xi_i \overset{\$}{\leftarrow} \mathbb{G}_2\right)_{i=0}^{u_h}, \quad \left(a_{t,i} \leftarrow T_i^c \prod_{j=0}^{n_{3,i}} t_{3,i,j}^{z_{\nu_{3,i,j}}} \prod_{j=0}^{n_{4,i}} \mathrm{e}(g_{4,i,j}, \xi_{\nu_{4,i,j}}) \prod_{j=0}^{n_{5,i}} \mathrm{e}(\zeta_{\nu_{5,i,j}}, h_{5,i,j})\right)_{i=0}^{n_t},$$

and finally outputs the transcript $\left(\left(\left(a_{g,i}\right)_{i=0}^{n_g}, \left(a_{h,i}\right)_{i=0}^{n_h}, \left(a_{t,i}\right)_{i=0}^{n_t}\right), c, \left(\left(z_i\right)_{i=0}^{n_g}, \left(\zeta_i\right)_{i=0}^{n_h}, \left(\xi_i\right)_{i=0}^{n_t}\right)\right)$.

C Perfect Zero-knowledge Proof of Knowledge

Perfect Zero-knowledge Proofs of Knowledge can be constructed by combining several Sigma protocols (see Appendix B) [CDM00]. We first recall the properties of perfect ZKPoK and then give the construction.

C.1 Properties of Perfect ZKPoK

A perfect ZKPoK protocol is run between two parties $\mathcal{P}$ and $\mathcal{V}$, which we model as PPT interactive machines. The following properties hold:

4-move Let $¢_{\mathcal{P}}$ and $¢_{\mathcal{V}}$ be some random coins, let $A(y,x)$ be the relation associated with the ZKPoK, then $\mathsf{ZKPoK}(\mathcal{P} \leftrightarrow \mathcal{V})$ consists of the following deterministic polynomial-time algorithms: $\mathcal{V}(y, ¢_{\mathcal{V}}) \to m_1$, $\mathcal{P}(y, x, ¢_{\mathcal{P}}, m_1) \to m_2$, $\mathcal{V}(y, ¢_{\mathcal{V}}, m_1, m_2) \to m_3$, $\mathcal{P}(y, x, ¢_{\mathcal{P}}, m_1, m_2, m_3) \to m_4$, $\mathcal{V}(y, m_1, m_2, m_3, m_4) \to b$. We also write $\mathrm{Out}_{\mathcal{V}}(\mathcal{P} \leftrightarrow \mathcal{V}) \overset{\text{def}}{=} b$.

Perfect completeness If $A(y,x) = 1$ then $\Pr_{¢_{\mathcal{V}}, ¢_{\mathcal{P}}}[\mathrm{Out}_{\mathcal{V}}(\mathcal{P} \leftrightarrow \mathcal{V}) \overset{\$?}{=} 1] = 1$.

Computational soundness There exists a PPT algorithm $\mathcal{E}$ (extractor) that, given black-box access to a PPT interactive machine $\mathcal{P}'$ (which may or may not follow the protocol honestly), can extract the witness x from $\mathcal{P}'$ with a probability larger than $\mathcal{P}'$ can convince the honest verifier $\mathcal{V}$ minus a negligible knowledge error μ:

$$\Pr[\mathcal{E}(\mathcal{P}') \xrightarrow{\$} x', A(y,x') \overset{?}{=} 1] \geq \Pr[\mathrm{Out}_{\mathcal{V}}(\mathcal{P}' \leftrightarrow \mathcal{V}) \overset{\$?}{=} 1] - \mu, \qquad \text{and } \mu = \mathit{negl}.$$

Usually we have that $\mu = 1/p$ (knowledge error is the inverse of the total number of challenges).

Perfect zero-knowledge There exists a PPT algorithm $\mathcal{S}$ (simulator) that, given black-box access to a PPT interactive machine $\mathcal{V}'$ (which may or may not follow the protocol honestly), outputs a transcript that is perfectly indistinguishable from the transcript produced by $\mathcal{V}' \leftrightarrow \mathcal{P}$ (with the honest prover).

C.2 Extensions to Sigma Protocols

C.2.1 Disjunctions of Sigma Protocols

Given two sigma protocols $\Sigma_1(\mathcal{P}_1 \leftrightarrow \mathcal{V}_1)$ and $\Sigma_2(\mathcal{P}_2 \leftrightarrow \mathcal{V}_2)$ for relations $A_1(y_1, x_1)$ and $A_2(y_2, x_2)$ it is possible to construct a third protocol $\Sigma_{\mathrm{OR}}(\mathcal{P}_{\mathrm{OR}} \leftrightarrow \mathcal{V}_{\mathrm{OR}}) \overset{\text{def}}{=} \Sigma_1 \vee \Sigma_2$ for the relation $A_{\mathrm{OR}}(y_{\mathrm{OR}}, x_{\mathrm{OR}}) \overset{\text{def}}{=} A_1(y_1, x_1) \vee A_2(y_2, x_2)$, i.e., where $\mathcal{P}_{\mathrm{OR}}$ proves that he either knows a witness for the first protocol, or a witness for the second protocol.

The construction is deceptively simple: Σ_1 and Σ_2 are run concurrently, except that $\mathcal{V}_{\mathrm{OR}}$ sends only a single challenge c_{OR} in the second move. $\mathcal{P}_{\mathrm{OR}}$ may then choose c_1 and c_2 such that $c_1 + c_2 = c_{\mathrm{OR}}$. Roughly speaking, this allows him to run the simulator of the sigma protocol he doesn't know the witness for, and adapt the challenge accordingly; however no such shortcut exists for the other sigma protocol, which he must therefore perform honestly [CS97a].

Without loss of generality, let's assume that $\mathcal{P}_{\mathrm{OR}}$ knows the witness x_1. The OR-protocol runs as follows [CS97a, CDM00]:

(1) $\mathcal{P}_{\mathrm{OR}}$ runs $\mathcal{P}_1$ to generate the commitment a_1, and runs the simulator of Σ_2 to get a transcript $\big(a_2, c_2, z_2\big)$. He sends a_1 and a_2 to $\mathcal{V}_{\mathrm{OR}}$.

(2) $\mathcal{V}_{\mathrm{OR}}$ generates $c \xleftarrow{\$} \mathbb{Z}_p$ and sends it to $\mathcal{P}_{\mathrm{OR}}$.

(3) $\mathcal{P}_{\mathrm{OR}}$ sets $c_1 \leftarrow (c - c_2)$ and runs $\mathcal{P}_1$ to generate the opening z_1. He sends c_1, c_2, z_1, z_2 over to $\mathcal{V}_{\mathrm{OR}}$.

(Check) $\mathcal{V}_{\mathrm{OR}}$ checks that $c_1 + c_2 = c$ and runs $\mathcal{V}_1$ and $\mathcal{V}_2$ to check the transcripts $\big(a_1, c_1, z_1\big)$ and $\big(a_2, c_2, z_2\big)$. If both return 1, then $\mathcal{V}_{\mathrm{OR}}$ does as well.

C.2.2 Sigma Protocol of Commitment Relationship

Given a transcript (a_3, c_3, z_3) for a sigma protocol $\Sigma_3(\mathcal{P}_3 \leftrightarrow \mathcal{V}_3)$ for relation $A(y_3, x_3)$, it is possible to construct another sigma protocol $\Sigma_4(\mathcal{P}_4 \leftrightarrow \mathcal{V}_4)$ for a relation $A(y_4 \overset{\text{def}}{=} \mathsf{commit\_rel}(y_3, a_3), x_4 \overset{\text{def}}{=} (c_3, z_3))$, where $\mathcal{P}_4$ proves he knows the opening z_3 and the challenge c_3 for the commitment a_3 ($\mathcal{P}_4$ does not need to know the witness x_3).

We work with Σ_3 of the form:

$$\Sigma_3\Big\{\Big(\big(\boxed{x_i} \in \mathbb{Z}_p\big)_{i=0}^{u_x}, \big(\boxed{\gamma_i} \in \mathbb{G}_1\big)_{i=0}^{u_g}, \big(\boxed{\eta_i} \in \mathbb{G}_2\big)_{i=0}^{u_h}\Big) : \quad \bigwedge_{i=0}^{n_g} \Big(G_i = \prod_{j=0}^{n_{1,i}} g_{1,i,j}^{\boxed{x_{\nu_{1,i,j}}}}\Big)$$
$$\bigwedge_{i=0}^{n_h} \Big(H_i = \prod_{j=0}^{n_{2,i}} h_{2,i,j}^{\boxed{x_{\nu_{2,i,j}}}}\Big) \bigwedge_{i=0}^{n_t} \Big(T_i = \prod_{j=0}^{n_{3,i}} t_{3,i,j}^{\boxed{x_{\nu_{3,i,j}}}} \prod_{j=0}^{n_{4,i}} \mathrm{e}(g_{4,i,j}, \boxed{\eta_{\nu_{4,i,j}}}) \prod_{j=0}^{n_{5,i}} \mathrm{e}(\boxed{\gamma_{\nu_{5,i,j}}}, h_{5,i,j})\Big)\Big\}.$$

We implicitly define $\mathsf{commit\_rel}$ as follows. The sigma protocol Σ_4 associated to the commitment relationship for the transcript $\big(a_3, c_3, z_3\big)$ of Sigma protocol Σ_3 is:

$$\Sigma_4\Big\{\Big(\boxed{c_3}, \big(\boxed{\tilde{z}_i} \in \mathbb{Z}_p\big)_{i=0}^{u_x}, \big(\boxed{\zeta_i} \in \mathbb{G}_1\big)_{i=0}^{u_g}, \big(\boxed{\xi_i} \in \mathbb{G}_2\big)_{i=0}^{u_h}\Big) : \quad \bigwedge_{i=0}^{n_g} \Big(a_{3,g,i} = G_i^{\boxed{c_3}} \prod_{j=0}^{n_{1,i}} g_{1,i,j}^{\boxed{\tilde{z}_{\nu_{1,i,j}}}}\Big)$$
$$\bigwedge_{i=0}^{n_h} \Big(a_{3,h,i} = H_i^{\boxed{c_3}} \prod_{j=0}^{n_{2,i}} h_{2,i,j}^{\boxed{\tilde{z}_{\nu_{2,i,j}}}}\Big) \bigwedge_{i=0}^{n_t} \Big(a_{3,t,i} = T_i^{\boxed{c_3}} \prod_{j=0}^{n_{3,i}} t_{3,i,j}^{\boxed{\tilde{z}_{\nu_{3,i,j}}}} \prod_{j=0}^{n_{4,i}} \mathrm{e}(g_{4,i,j}, \boxed{\xi_{\nu_{4,i,j}}}) \prod_{j=0}^{n_{5,i}} \mathrm{e}(\boxed{\zeta_{\nu_{5,i,j}}}, h_{5,i,j})\Big)\Big\},$$

where $a_3 \overset{\text{def}}{=} \Big(\big(a_{3,g,i} \in \mathbb{G}_1\big)_{i=0}^{n_g}, \big(a_{3,h,i} \in \mathbb{G}_2\big)_{i=0}^{n_h}, \big(a_{3,t,i} \in \mathbb{G}_{\mathrm{T}}\big)_{i=0}^{n_t}\Big)$ was the commitment message, $\boxed{c_3}$ the challenge, and where $\boxed{z_3} \overset{\text{def}}{=} \Big(\big(\boxed{\tilde{z}_i} \in \mathbb{Z}_p\big)_{i=0}^{u_x}, \big(\boxed{\zeta_i} \in \mathbb{G}_1\big)_{i=0}^{u_g}, \big(\boxed{\xi_i} \in \mathbb{G}_2\big)_{i=0}^{u_h}\Big)$ was the opening in the transcript of Σ_3.

C.3 The Construction

We will describe the conceptually simpler 6-move variant of perfect ZKPoK, which runs in two parts of three moves each. The 4-move protocol is obtained by merging the second and third message of part one with the first and second message of part two [CDM00].

Let $\Sigma_3(\mathcal{P}_3 \leftrightarrow \mathcal{V}_3)$ be a sigma protocol for the relation $A_3(x_3, w_3)$. We now show how to construct a perfect zero-knowledge proof of knowledge $\mathsf{ZKPoK}(\mathcal{P} \leftrightarrow \mathcal{V})$ for the same relation:

(1–3) $\mathcal{V}$ runs the simulator of Σ_3 to generate a transcript $\big(a_3, c_3, z_3\big)$. Let $\Sigma_4(\mathcal{P}_4 \leftrightarrow \mathcal{V}_4)$ be the sigma protocol associated to the commitment relationship for that transcript. We have $A_4(\mathsf{commit\_rel}(a_3, x_3), (c_3, z_3))$. $\mathcal{V}$ sends a_3 to $\mathcal{P}$. $\mathcal{V}$ and $\mathcal{P}$ then run $\mathcal{P}_4$ and $\mathcal{V}_4$ respectively, so that $\mathcal{V}$ can prove he knows the transcript. The protocol aborts if $\mathcal{V}_4$ returns 0.

(4–6) Let $\Sigma_{\mathrm{OR}}(\mathcal{P}_{\mathrm{OR}}, \mathcal{V}_{\mathrm{OR}}) \leftarrow \Sigma_3 \vee \Sigma_4$. $\mathcal{P}$ and $\mathcal{V}$ now run respectively $\mathcal{P}_{\mathrm{OR}}$ and $\mathcal{V}_{\mathrm{OR}}$. The protocol completes successfully if $\mathcal{V}_{\mathrm{OR}}$ returns 1.

D Groth-Sahai Zero-knowledge Proofs

Let $y \stackrel{\text{def}}{=} \left\{n \in \mathbb{N}, \left(n_i \in \mathbb{N}, G_i \in \mathbb{G}_1, \left(\nu_{i,j} \in \mathbb{N}_{u+1}, g_{i,j} \in \mathbb{G}_1\right)_{j=0}^{n_i}\right)_{i=0}^{n}\right\}$ and let $x \stackrel{\text{def}}{=} \left\{\boxed{x_i}\right\}_{i=0}^{u}$ and let $A(y, x)$ be the following statement:

$$\bigwedge_{i=0}^{n} \left(G_i = \prod_{j=0}^{n_i} g_{i,j}^{\boxed{x_{\nu_{i,j}}}}\right).$$

A Groth Sahai non-interactive zero-knowledge proof allows a prover $\mathcal{P}$ to generate a proof π certifying that there exists an x such as $A(y, x)$ is satisfied (on public input y), without revealing x. Groth-Sahai proofs can also be used to prove more general statements than the one above [GS08], but this is out of the scope of this document.

D.1 Types of Common Reference String (CRS)

Two flavors of CRS are defined for GS proofs:

- Either a *binding CRS* (also called *real CRS*). The GS proofs will then be perfectly binding and computationally hiding. This is the type of CRS that is used with honest GS proofs. In a real scheme, all participants rely on a trusted third party to generate a binding CRS without trapdoor for them.

- Or a *hiding CRS* (also called *simulated CRS*). The GS proofs will be perfectly hiding and computationally binding. This type of CRS is used only when proving security and never in a real scheme.

These two kinds of CRS are computationally indistinguishable from each other under the SXDH assumption.

D.2 Security Properties

We now define some properties satisfied by GS-proofs [GS08]. We can construct GS proofs that are perfectly complete, sound and zero knowledge for the limited subset of equation types we use.[36]

A non-interactive proof system for a relation $A(y, x)$ consists of four PPT algorithms: A bilinear group setup algorithm $\mathcal{G}$, a CRS generation algorithm $\mathcal{K}$, an honest prover $\mathcal{P}$ and an honest verifier $\mathcal{V}$.

For GS-proofs, $\mathcal{G}$ will output a description of a bilinear group gk given a security parameter κ. $\mathcal{K}$ takes as input gk and outputs a binding CRS $\mathfrak{U}$. The prover $\mathcal{P}$ takes as input $(gk, \mathfrak{U}, y, x)$ and outputs a proof π. The verifier $\mathcal{V}$ takes as input $(gk, \mathfrak{U}, y, \pi)$ and outputs 1 if he accepts the proof or 0 if he rejects the proof.

In what follows, the bilinear group setup $gk \stackrel{\$}{\leftarrow} \mathcal{G}(\mathbf{1}^\kappa)$ is implied to save space.

Perfect Completeness The proof made by an honest prover will always be accepted by an honest verifier. Formally for all unbounded adversaries $\mathcal{A}$ [GS08]:

$$\Pr\left[\mathfrak{U} \stackrel{\$}{\leftarrow} \mathcal{K}(gk),\quad (y, x) \stackrel{\$}{\leftarrow} \mathcal{A}(gk, \mathfrak{U}),\quad \pi \stackrel{\$}{\leftarrow} \mathcal{P}(gk, \mathfrak{U}, y, x),\quad \mathcal{V}(gk, \mathfrak{U}, y, \pi) \stackrel{\$?}{=} 1 \;\middle|\; A(y, x) = 1\right] = 1.$$

[36]For general GS proofs, it might not possible to achieve the Zero-Knowledge property, but only a weaker notion of Composable Witness Indistinguishability.

Perfect Soundness The proof made by a dishonest prover for an unsatisfiable statement will never be accepted by an honest verifier. Formally for all unbounded adversaries $\mathcal{A}$ [GS08]:

$$\Pr\left[\mathfrak{U} \xleftarrow{\$} \mathcal{K}(gk), \quad (y,\pi) \xleftarrow{\$} \mathcal{A}(gk,\mathfrak{U}), \quad \mathcal{V}(gk,\mathfrak{U},y,\pi) \overset{\$?}{=} 0 \;\middle|\; \nexists x : A(y,x)=1\right] = 1.$$

Composable Non-Interactive Zero Knowledge (NIZK) Informally, (Composable) NIZK means that a PPT simulator which has access to a trapdoor $\mathfrak{t}$ of the CRS (or who can program the CRS) can produce accepting proofs for any statement for which he does not know the witness.

Formally there exists a PPT CRS simulator $\mathcal{S}_1$ (generating a hiding CRS and a trapdoor) and a PPT simulator $\mathcal{S}_2$ such that for all PPT adversaries $\mathcal{A}$ [GS08]:

$\mathcal{A}$ cannot distinguish a binding CRS from a hiding CRS:

$$\left|\Pr\left[\mathfrak{U} \xleftarrow{\$} \mathcal{K}(gk), \quad \mathcal{A}(gk,\mathfrak{U}) \overset{\$?}{=} 1\right] - \Pr\left[\{\mathfrak{U},\mathfrak{t}\} \xleftarrow{\$} \mathcal{S}_1(gk), \quad \mathcal{A}(gk,\mathfrak{U}) \overset{\$?}{=} 1\right]\right| = negl,$$

and $\mathcal{A}$ can't determine if the proof was simulated when using a hiding CRS, *even if given access to the trapdoor to the CRS*:

$$\{\mathfrak{U},\mathfrak{t}\} \xleftarrow{\$} \mathcal{S}_1(gk), \quad (y,x) \xleftarrow{\$} \mathcal{A}(gk,\mathfrak{U},\mathfrak{t}) \text{ such that } A(y,x)=1,$$
$$\Pr\left[\pi \xleftarrow{\$} \mathcal{P}(gk,\mathfrak{U},y,x) \quad \mathcal{A}(\pi) \overset{\$?}{=} 1\right] = \Pr\left[\pi \xleftarrow{\$} \mathcal{S}_2(gk,\mathfrak{U},y,\mathfrak{t}) \quad \mathcal{A}(\pi) \overset{\$?}{=} 1\right].$$

Composable NIZK implies the standard notion of zero-knowledge.

D.3 Construction

D.3.1 Transformation

Let $1_{\mathbb{G}_1} \overset{\text{def}}{=} g_1^0$ be the neutral element of $\mathbb{G}_1$. The first step of the GS proof is to transform the statement $\bigwedge_{i=1}^n G_i = \prod_{j=0}^{n_i} g_{i,j}^{\boxed{x_{\nu_{i,j}}}}$ into the equivalent statement $\bigwedge_{i=1}^n 1_{\mathbb{G}_1} = G_i^{\boxed{\varphi}} \prod_{j=0}^{n_i} g_{i,j}^{\boxed{x_{\nu_{i,j}}}}$. To that effect we introduce a dummy unknown $\boxed{\varphi} = -1$ (the prover will publish the random value s_{u+1} he uses to commit to $\boxed{\varphi}$).

Roughly speaking, the simulator can achieve zero knowledge in the hiding setup by setting all unknowns—including $\boxed{\varphi}$—to zero when computing the proof (the statement will thus trivially be satisfied), and publish a value of s_{u+1} that tricks the verifier into thinking that the statement was proven for $\boxed{\varphi} = -1$ [GS08, Section 8].

D.3.2 CRS Setup

Binding CRS Setup The honest CRS generation algorithm $\mathcal{K}$ computes:

$$\mathfrak{a},\mathfrak{t} \xleftarrow{\$} \mathbb{Z}_p^*, \qquad \mathfrak{U}_{1,2} \leftarrow g_2^{\mathfrak{a}}, \qquad \mathfrak{U}_{2,1} \leftarrow g_2^{\mathfrak{t}}, \qquad \mathfrak{U}_{2,2} \leftarrow g_2^{\mathfrak{a}\mathfrak{t}}.$$

The CRS is $\{\mathfrak{U}_{1,2},\mathfrak{U}_{2,1},\mathfrak{U}_{2,2}\}$. The random elements $\{\mathfrak{a},\mathfrak{t}\}$ are discarded.

Hiding CRS Setup The CRS Simulator $\mathcal{S}_1$ computes:

$$\mathfrak{a},\mathfrak{t} \xleftarrow{\$} \mathbb{Z}_p^*, \qquad \mathfrak{U}_{1,2} \leftarrow g_2^{\mathfrak{a}}, \qquad \mathfrak{U}_{2,1} \leftarrow g_2^{\mathfrak{t}}, \qquad \mathfrak{U}_{2,2} \leftarrow g_2^{\mathfrak{a}\mathfrak{t}-1}.$$

The CRS is $\{\mathfrak{U}_{1,2},\mathfrak{U}_{2,1},\mathfrak{U}_{2,2}\}$. The trapdoor is $\{\mathfrak{a},\mathfrak{t}\}$. Note that this type of CRS is only used in the security proofs, and never in the real scheme.

D.3.3 Computing the Proof

Let's define $x_{u+1} \overset{\text{def}}{=} \varphi \overset{\text{def}}{=} -1$ and $\forall i \in \mathbb{N}_{n+1} : g_{i,n_i+1} \overset{\text{def}}{=} G_i$. The prover $\mathcal{P}$ computes the proof as follows:

$$\big(s_i \overset{\$}{\leftarrow} \mathbb{Z}_p\big)_{i=0}^{u+1}, \quad \big(d_{i,1} \leftarrow \mathfrak{U}_{2,1}^{x_i} g_2^{s_i}\big)_{i=0}^{u+1}, \quad \big(d_{i,2} \leftarrow (\mathfrak{U}_{2,2} g_2)^{x_i} \mathfrak{U}_{1,2}^{s_i}\big)_{i=0}^{u+1}, \quad \big(\theta_i \leftarrow \prod_{j=0}^{n_i+1} g_{i,j}^{s_{\nu_{i,j}}}\big)_{i=0}^{n}.$$

The proof that $\mathcal{P}$ publishes is $\big\{\big(d_{i,1}, d_{i,2}\big)_{i=0}^{u}, s_{u+1}, \big(\theta_i\big)_{i=0}^{n}\big\}$.

D.3.4 Verification

The verifier $\mathcal{V}$ first recomputes the commitments to $\varphi = -1$:

$$d_{u+1,1} \leftarrow \mathfrak{U}_{2,1}^{-1} g_2^{s_{u+1}}, \qquad\qquad d_{u+1,2} \leftarrow (\mathfrak{U}_{2,2} g_2)^{-1} \mathfrak{U}_{1,2}^{s_{u+1}}.$$

He returns 1 if and only if the following statement is satisfied [GSW10]: [37]

$$\bigwedge_{i=0}^{n} \Big(\prod_{j=0}^{n_i+1} \mathrm{e}(g_{i,j}, d_{\nu_{i,j},1}) \overset{?}{=} \mathrm{e}(\theta_i, g_2) \wedge \prod_{j=0}^{n_i+1} \mathrm{e}(g_{i,j}, d_{\nu_{i,j},2}) \overset{?}{=} \mathrm{e}(\theta_i, \mathfrak{U}_{1,2}) \Big).$$

E Structure-preserving Signatures

In this section we recall the construction of structure-preserving signatures for vectors of $n \in \mathbb{N}^*$ group elements by Abe et al. [AFG+10, Section 5] [AHO10].

We show here how to sign a message tuple in $\mathbb{G}_2$. For messages in $\mathbb{G}_1$ we simply need to switch the groups $\mathbb{G}_1$ and $\mathbb{G}_2$ in everything that follows.

E.1 Existential Unforgeability under Chosen-message Attack

A signature scheme is existentially unforgeable under chosen-message attack (EUF-CMA) if no PPT adversary $\mathcal{A}$ can win the following game with non-negligible advantage:

(1) The challenger generates the signing keys (for message tuples of size $n \in \mathbb{N}^*$) and hands over the verification key *vk* to $\mathcal{A}$.

(2) $\mathcal{A}$ selects a message tuple $M = \big(m_1 \in \mathbb{G}_2, \ldots, m_n \in \mathbb{G}_2\big)$ of his choice. The challenger then generates the signature σ on that message tuple and hands it to $\mathcal{A}$. This step may be repeated polynomially many times.

(3) $\mathcal{A}$ outputs a message tuple M' and a signature σ'. $\mathcal{A}$ wins if σ' is a valid signature on M', and M' is different from all message tuples M he has queried for in step (2).

Abe et al.'s signature scheme is EUF-CMA provided the ℓ-SFP assumption holds in $\mathbb{G}_1$, $\mathbb{G}_2$ (with ℓ being the number of queries the adversary makes).

[37] We take the verification equations from Ghadafi et al.[GSW10], as the equations in the original paper [GS08] are not correct.

E.2 Key Generation (SigKeyGen)

To create a signing and verification key pair for message tuples of size n, one computes the following:

$$g_R \xleftarrow{\$} \mathbb{G}_1^*, \quad \gamma_Z, \alpha \xleftarrow{\$} \mathbb{Z}_p^*, \quad \big(\gamma_{M,i} \xleftarrow{\$} \mathbb{Z}_p^*\big)_{i=1}^n, \quad \big(g_{M,i} \leftarrow g_R^{\gamma_{M,i}}\big)_{i=1}^n, \quad g_Z \leftarrow g_R^{\gamma_Z}, \quad A \leftarrow \mathrm{e}(g_R, g_2^\alpha),$$
$$f_U \xleftarrow{\$} \mathbb{G}_1^*, \quad \delta_Z, \beta \xleftarrow{\$} \mathbb{Z}_p^*, \quad \big(\delta_{M,i} \xleftarrow{\$} \mathbb{Z}_p^*\big)_{i=1}^n, \quad \big(f_{M,i} \leftarrow f_U^{\delta_{M,i}}\big)_{i=1}^n, \quad f_Z \leftarrow f_U^{\delta_Z}, \quad B \leftarrow \mathrm{e}(f_U, g_2^\beta).$$

The verification key is $vk \overset{\text{def}}{=} \Big\{g_Z, f_Z, g_R, f_U, \big(g_{M,i}, f_{M,i}\big)_{i=1}^n, A, B\Big\}$.
The signing key is $sgk \overset{\text{def}}{=} \{\alpha, \beta, \gamma_Z, \delta_Z, \big(\gamma_{M,i}, \delta_{M,i}\big)_{i=1}^n\}$.

E.3 Signing (Sign)

To sign a message tuple $M = \big(m_1 \in \mathbb{G}_2, \ldots, m_n \in \mathbb{G}_2\big)$ we proceed as follows:

$$\zeta, \rho, \tau \xleftarrow{\$} \mathbb{Z}_p^*, \qquad R \leftarrow g_2^{\rho - \gamma_Z \zeta} \prod_{i=1}^n m_i^{-\gamma_{M,i}}, \qquad S \leftarrow g_R^\tau, \qquad T \leftarrow g_2^{(\alpha-\rho)\tau^{-1}}, \qquad Z \leftarrow g_2^\zeta,$$
$$\varphi, \omega \xleftarrow{\$} \mathbb{Z}_p^*, \qquad U \leftarrow g_2^{\varphi - \delta_Z \zeta} \prod_{i=1}^n m_i^{-\delta_{M,i}}, \qquad V \leftarrow f_U^\omega, \qquad W \leftarrow g_2^{(\beta-\varphi)\omega^{-1}}.$$

The signature is $\sigma \overset{\text{def}}{=} \{Z, R, S, T, U, V, W\}$.

E.4 Verification (SigVerify)

A signature is correct if the following two equations hold:

$$A \overset{?}{=} \mathrm{e}(g_Z, Z)\, \mathrm{e}(g_R, R)\, \mathrm{e}(S, T) \prod_{i=1}^n \mathrm{e}(g_{M,i}, m_i),$$
$$B \overset{?}{=} \mathrm{e}(f_Z, Z)\, \mathrm{e}(f_U, U)\, \mathrm{e}(V, W) \prod_{i=1}^n \mathrm{e}(f_{M,i}, m_i).$$

E.5 Re-randomizing a Signature (SigRerand)

To partially re-randomize a signature σ (for the message tuple m) we proceed as follows:

$$\rho \xleftarrow{\$} \mathbb{Z}_p^*, \qquad \gamma \xleftarrow{\$} \mathbb{Z}_p^*, \qquad R' \leftarrow RT^\rho, \qquad S' \leftarrow (Sg_R^{-\rho})^\gamma, \qquad T' \leftarrow T^{\gamma^{-1}},$$
$$\tau \xleftarrow{\$} \mathbb{Z}_p^*, \qquad \omega \xleftarrow{\$} \mathbb{Z}_p^*, \qquad U' \leftarrow UW^\tau, \qquad V' \leftarrow (Vf_U^{-\tau})^\omega, \qquad W' \leftarrow W^{\omega^{-1}}.$$

The re-randomized signature $\sigma' \overset{\text{def}}{=} \{Z, R', S', T', U', V', W'\}$ is also a valid signature for the message tuple m.

The variables S, T, V, W are information theoretically independent of Z and m. One can therefore reveal these values in a Zero-Knowledge proof.

E.6 ZKPoK of Private Key (**SignKeyZKPoK**)

We define the following shorthand notation to prove knowledge of the signing key corresponding to a verification key: $\mathsf{ZKPoK}_{\mathrm{SigKey}}(vk)$. By this we mean:

$$\mathsf{ZKPoK}\Bigg\{\Big(\boxed{\alpha}, \boxed{\beta}, \boxed{\gamma_Z}, \boxed{\delta_Z}, \big(\boxed{\gamma_M}, \boxed{\delta_M}\big)_{i=1}^{n}\Big) : A = \mathrm{e}(g_R, g_2)^{\boxed{\alpha}} \wedge B = \mathrm{e}(f_U, g_2)^{\boxed{\beta}} \wedge$$
$$g_Z = g_R^{\boxed{\gamma_Z}} \wedge f_Z = f_U^{\boxed{\delta_Z}} \bigwedge_{i=1}^{n} \Big(g_{M,i} = g_R^{\boxed{\gamma_{M,i}}} \wedge f_{M,i} = f_U^{\boxed{\delta_{M,i}}} \Big) \Bigg\}.$$