An extended abstract of this paper appears at the 16th ACM Conference on Computer and Communications Security (ACM CCS 2009). This is the full version.

Oblivious Transfer with Access Control

Jan Camenisch[1] Maria Dubovitskaya[2,3] Gregory Neven[1]

[1] IBM Research – Zurich
jca,nev@zurich.ibm.com
[2] IBM Russian Systems and Technology Laboratory
[3] Moscow Engineering Physics Institute (State University)
maria@ru.ibm.com

Abstract

We present a protocol for anonymous access to a database where the different records have different access control permissions. These permissions could be attributes, roles, or rights that the user needs to have in order to access the record. Our protocol offers maximal security guarantees for both the database and the user, namely (1) only authorized users can access the record; (2) the database provider does not learn which record the user accesses; and (3) the database provider does not learn which attributes or roles the user has when she accesses the database.

We prove our protocol secure in the standard model (i.e., without random oracles) under the bilinear Diffie-Hellman exponent and the strong Diffie-Hellman assumptions.

1 Introduction

More and more transactions in our daily life are performed electronically. People enter their credentials online and into various databases and disclose their personal information to different organisations with the belief that small amounts of information cannot reveal enough about them to impact them in a negative way. When using the internet extensively however, they can give away much more information about themselves than they may care to admit.

Also to protect sensitive information such as medical or financial data we need to provide strong access control to be sure that only those people who have the necessary permissions can access it. But statistics about what sort of data people query also reveals a lot of information about them.

It is possible to build a complete picture of someone's movements, transactions, locations and relationships from the trail left from interaction with websites and various databases. So personal security has become a serious issue.

To protect the users' privacy, it is important that all electronic transactions can be performed without revealing more personal information than is absolutely necessary. In this paper we consider the case of access to a database where the different records in the database have different access control conditions. These conditions could be certain attributes, roles, or rights that a user needs

to have to access the records. The assigning of attributes to users is done by a separate entity called the issuer, external to the database. To provide the maximal amount of privacy, a protocol is required such that:

- Only users satisfying the access conditions for a record can access that record;
- The service (database) provider does not learn which record a user accesses;
- The service (database) provider shall not learn which attributes, roles, etc. a user has when she accesses a record, i.e., access shall be completely anonymous, nor shall it learn which attributes the user was required to have to access the record.

One real-life example where such a protocol is important are DNA databases, containing information about the purpose of each gene. Such databases are extremely valuable and thus there are not sold on a whole, but rather customers are charged per access to the database. On the other hand, the particular DNA sequences accessed by a customer reveal a lot of information about her interests, e.g., for which disease it is developing medication. Moreover, it is quite likely that subscription prices vary with the different species. Using our protocol, the database can charge different rates for the DNA sequences of mice and apes, without forcing its customers to reveal which species they're interested in.

Other examples of databases where users have an interest to keep their queries hidden are stock quotes, since they can reveal information about their investment strategy, and patent search, since they can reveal sensitive business information. Our protocol directly addresses these problems and provides a practical solution for it.

1.1 Construction Overview

We now describe the main ideas underlying our protocol. We build upon the oblivious transfer protocol by Camenisch, Neven, and Shelat [14] which we describe first. In their scheme, the server first encrypts each record with a unique key and publishes these encryptions. The encryption key is derived from the index of the record and a secret of the database server. Although the secret of the database is the same for all record keys, it is not possible to derive the encryption key for one record from that of another record. Thus, to be able to access a record, a user needs to retrieve the corresponding key from the server. To this end, Camenisch et al. give a protocol ensuring that 1) the user can retrieve exactly one key per protocol run and 2) the server does not learn which key the user obtained.

The main ideas of our scheme are as follows. First, we issue anonymous credentials [19, 20, 26, 22, 8, 31, 11] to a user, each certifying a *category* of records the user is allowed to access. Recall that anonymous credentials allow the user to later prove that she possesses a credential without revealing any other information whatsoever. We note that the name "category" is inspired by the different data categories that a user is allowed to access. However, the category could just as well encode the right, role, or attribute that a user needs to have in order to access a record. In the following we will only use the word category, however. Also, note that if a record has several categories attached to it, then the user must have a credential for *all* these categories, basically implementing an AND condition. If one would want to specify an OR condition, one could duplicate the record in the database with a second set of categories.

To allow the user oblivious access a record for which she has the necessary credentials, we extend the Camenisch et al. oblivious transfer protocol in two ways: 1) The keys for a record are derived

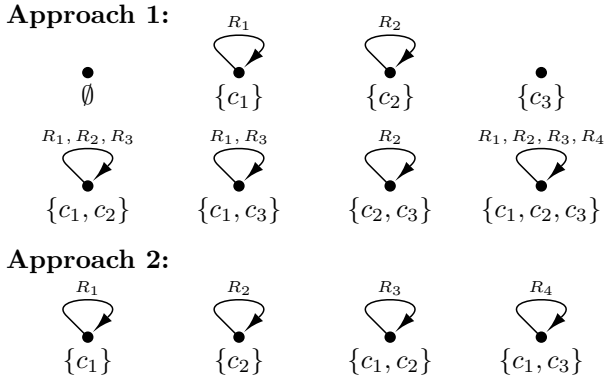

Figure 1: Access control graphs when implementing AC-OT using Coull et al.'s protcol for database $(R_1, \{c_1\}), (R_2, \{c_2\}), (R_3, \{c_1, c_2\}), (R_4, \{c_1, c_3\})$.

not only from the index of the record and the secret key of the server but also from the categories of the record. 2) We extend the protocol so that the user, while retrieving a key, gives a zero-knowledge proof of knowledge that she possess credentials on all the categories that are encoded into the key that she wants to retrieve. Using anonymous credentials and the specific features of the Camenisch et al. protocol, we can do this without letting the server learn the categories nor any other information about the key which the user obtains trough the protocol.

1.2 Related Work

There is of course a large body of works on oblivious transfer which per se offers users access to a database without the server learning the contents of the query. In its basic form, oblivious transfer puts no restrictions on which records a particular user can access, i.e., all users can access all records. There are a couple of papers that consider oblivious transfer with access control, each of them, however, aiming at a goal different from ours.

Aiello, Ishai, and Reingold [1] present *priced* oblivious transfer. Here, each record has attached a (possibly different) price. The user holds a (homomorphically) encrypted balance which is reduced with each transfer. Thus, the user can only retrieve records as long as her balance is positive. Another related flavor is *conditional* oblivious transfer, proposed by Di Crescenzo, Ostrovsky, and Rajagopolan [27], where access to a record is only granted if the user's secret satisfies some given predicate. However, none of these protocols offer anonymity to the users.

Herranz [30] proposes *restricted* oblivious transfer, which also protects each record with an access control policy. In his case the policy consists of a list saying which user has access to which record, and the user authenticates to the server openly. In contrast, our protocol employs a more powerful attribute-based access control paradigm, and guarantees user anonymity.

To the best of our knowledge, the only paper considering oblivious transfer with access control is the recent work by Coull, Green, and Hohenberger [23]. They propose a scheme for controlling access to records using state graphs. With each access a user transitions from one state to another, where the transition is defined by the index of the record the user has accessed. By restricting the between states, a user being in a particular state can only access the records corresponding to the possible transitions.

An exact comparison between our protocol and that of [23] depends on the particular access structure of the database and on how the AC-OT primitive is translated into a graph structure. In general however, our protocol is more efficient because it avoids re-issuing user credentials at each transfer. We discuss two ways of implementing AC-OT using Coull et al.'s protocol below.

One approach (see Approach 1 in Figure 1) could be to assign a state to each subset of categories that a user could have access to, with a self-loop for each record that can be accessed using this subset. When given access to a new category, the user is re-initialized in the state representing her new set of categories. For a database of N records and C different categories, this yields a graph of 2^C nodes and up to N edges per node, yielding an encrypted database size of $O(2^C N)$ using Coull et al.'s protocol, versus $O(N)$ using ours.

Another approach (Approach 2 in Figure 1) could be to assign a state to each subset of categories that appears as an access control list in the database, and a self-loop for each record having that access control list. Users are in multiple states simultaneously, namely in all those corresponding to access control policies that they satisfy. This yields a graph of up to N nodes and N edges total, reducing the encrypted database size to $O(N)$, but requires users to store up to $\min(2^C, N)$ different credentials, and partly destroys the intuitive aspect of using data categories for access control.

2 Definition of AC-OT

2.1 Overview

An oblivious transfer protocol with access control (AC-OT) is run between the following parties:

- users (U_1, ..., U_M) known by pseudonyms;
- an issuer I providing access credentials to users for the data categories that they are entitled to access;
- a database DB hosting the list of records and giving users access to those records that they are entitled to access.

In a nutshell, an oblivious transfer protocol with access control works as follows.

1. The issuer I generates his key pair for issuing credentials and publishes the public key as a system-wide parameter.
2. The database server initiates a database containing records protected by access control lists: generates the encrypted database and makes it available to all users, e.g. by posting it on a website.
3. Users contact the issuer to obtain credentials for the data categories that they want or are entitled to access.
4. When a user wants to access a record in the database, she proves to the database, in a zero-knowlege way, that she possesses credentials for all categories associated with this record. If she succeeds then she can decrypt that record, otherwise, she cannot. The database learns nothing about the index of the record that is being accessed, nor about the categories associated to the record.

2.2 Syntax

If $\kappa \in \mathbb{N}$, then 1^κ is the string consisting of κ ones. The empty string is denoted ε. If A is a randomized algorithm, then $y \xleftarrow{\$} \mathsf{A}(x)$ denotes the assignment to y of the output of A on input x when run with fresh random coins.

Unless noted, all algorithms are probabilistic polynomial-time (PPT) and we implicitly assume they take an extra parameter 1^κ in their input, where κ is a security parameter. A function $\nu : \mathbb{N} \to [0,1]$ is *negligible* if for all $c \in \mathbb{N}$ there exists a $\kappa_c \in \mathbb{N}$ such that $\nu(\kappa) < \kappa^{-c}$ for all $\kappa > \kappa_c$.

We consider a setting with one issuer, one database, and one or more users. Data categories are bit strings taken from the category universe $\mathcal{C} \subseteq \{0,1\}^*$. A database consists of a list of N couples $((R_1, ACL_1), \ldots, (R_N, ACL_N))$, containing database records $R_1, \ldots, R_N \in \{0,1\}^*$ and associated access control lists $ACL_1, \ldots, ACL_N \subseteq \mathcal{C}$. The semantics of the access control lists is that only users who have credentials for *all* data categories in ACL_i can access R_i. In other words, the access control list is a conjunction of keywords; disjunctions can be realized by letting the same record appear multiple times in the database. Finally, users interact with the database directly to obtain those records that they are entitled to receive.

An adaptive oblivious transfer protocol with access control (AC-OT) for category universe $\mathcal{C} \subseteq \{0,1\}^*$ is a tuple of polynomial-time algorithms and protocols $\mathcal{AC}\text{-}\mathcal{OT} = (\mathsf{ISetup}, \mathsf{Issue}, \mathsf{DBSetup}, \mathsf{Transfer})$.

- $\mathsf{ISetup}(\mathcal{C}) \xrightarrow{\$} (pk_\mathrm{I}, sk_\mathrm{I})$
  The issuer runs the randomized ISetup algorithm to generate a public key pk_I and corresponding secret key sk_I for security parameter κ and category universe $\mathcal{C}$. He publishes the public key as a system-wide parameter.

- Issue: Common input: pk_I, c
  Issuer input: sk_I
  User input: st_U
  User output: $cred_c$ or $\perp$, $st_\mathrm{U}{}'$

  A user obtains an access credential for data category $c \in \mathcal{C}$ by engaging in the Issue protocol with the issuer. The issuer's public key pk_I and the data category c are common inputs. The issuer uses his secret key sk_I as an input and the user possibly maintains his state st_U. At the end of the protocol, the user obtains the access credential $cred_c$ and the updated state $st_\mathrm{U}{}'$.

- $\mathsf{DBSetup}\big(pk_\mathrm{I}, DB = (R_i, ACL_i)_{i=1,\ldots,N}\big) \xrightarrow{\$} \big((pk_\mathrm{DB}, ER_1, \ldots, ER_N),\ sk_\mathrm{DB}\big)$
  To initiate a database containing records $R_1, \ldots, R_N$ protected by access control lists $ACL_1, \ldots, ACL_N$, the database server runs the $\mathsf{DBSetup}$ algorithm. This generates the encrypted database consisting of a public key pk_DB and encrypted records $ER_1, \ldots, ER_N$. The encrypted database is made available to all users, e.g. by posting it on a website.[1] The server keeps the secret key to the database sk_DB for itself.

[1] We assume that each user obtains a copy of the entire encrypted database. It is impossible to obtaining our strong privacy requirements with a single database server without running into either computation or communication complexity that is linear in the database size.

- Transfer: Common input: $pk_{\mathrm{I}}, pk_{\mathrm{DB}}$
  User input: $\sigma, ER_\sigma, ACL_\sigma, \{cred_c\}_{c \in ACL_\sigma}, st_{\mathrm{U}}$
  Database input: sk_{DB}
  User output: R_σ or $\perp$, $st_{\mathrm{U}}{}'$

  When the user wants to access a record in the database, she engages in a Transfer protocol with the database server. Common inputs are the issuer's public key pk_{I} and that of the database pk_{DB}. The user has as a secret input her selection index $\sigma \in \{1, \ldots, N\}$, the required credentials $cred_c$ for all $c \in ACL_\sigma$, and possibly state information st_{U}. The database server uses its secret key sk_{DB} as a private input. At the end of the protocol, the user obtains the database record R_σ or $\perp$ indicating failure, and updated state $st_{\mathrm{U}}{}'$.

We assume that all communication links are private. We also assume that the communication links between a user and the issuer are authenticated, so that the issuer always knows which user it is handing a credential to. The communication links between a user and the database are assumed to be anonymous however, so that the database does not know which user is making a record query. (Authenticated communication channels between users and the database would obviously ruin the strong anonymity properties of our protocol.)

2.3 Security

We define security of an AC-OT protocol through indistinguishability of a real-world and an ideal-world experiment as introduced by the UC framework [17, 18] and the reactive systems security models [32, 33]. The definitions we give, however, do not entail all formalities necessary to fit one of these frameworks; our goal here is solely to prove security of our scheme.

We summarize the ideas underlying these models. In the real world there are a number of players, who run some cryptographic protocols with each other, an adversary A, who controls some of the players, and an environment $\mathcal{E}$. The environment provides the inputs to the honest players and receives their outputs and interacts arbitrarily with the adversary. The dishonest players are subsumed into the adversary.

In the ideal system, we have the same players. However, they do not run any cryptographic protocols but send all their inputs to and receive all their outputs from an ideal all-trusted party T. This party computes the output of the players from their inputs, i.e., applies the functionality that the cryptographic protocol(s) are supposed to realize. The environment again provides the inputs to and receives the output from the honest players, and interacts arbitrarily with the adversary controlling the dishonest players.

A (set of) cryptographic protocol(s) is said to securely implement a functionality if for every real-world adversary A and every environment $\mathcal{E}$ there exists an ideal-world simulator A' controlling the same parties in the ideal world as A does in the real world such that the environment cannot distinguish whether it is run in the real world interacting with A or whether it is run in the ideal world interacting with the simulator A'.

Definition 2.1 Let $\mathbf{Real}_{\mathcal{E},\mathsf{A}}(\kappa)$ denote the probability that $\mathcal{E}$ outputs 1 when run in the real world with A and $\mathbf{Ideal}_{\mathcal{E},\mathsf{A}'}(\kappa)$ denotes the probability that $\mathcal{E}$ outputs 1 when run in the ideal world interacting with A', then the (set of) cryptographic protocols is said to securely implement the functionality T if

$$\mathbf{Real}_{\mathcal{E},\mathsf{A}}(\kappa) - \mathbf{Ideal}_{\mathcal{E},\mathsf{A}'}(\kappa)$$

is a negligible function in κ.

THE REAL WORLD. We first describe how the real world algorithms presented in §2.2 are orchestrated when all participants are honest, i.e., honest real-world users $\mathsf{U}_1, \ldots \mathsf{U}_M$, an honest issuer I, and an honest database DB. Parties controlled by the real-world adversary A can arbitrarily deviate from the behavior described below.

All begins with the issuer I generating a key pair $(pk_{\mathrm{I}}, sk_{\mathrm{I}}) \xleftarrow{\$} \mathsf{Issue}(1^\kappa, \mathcal{C})$ and sending pk_{I} to all users $\mathsf{U}_1, \ldots, \mathsf{U}_M$ and the database DB.

When the environment $\mathcal{E}$ sends a message $(\texttt{initdb}, DB = (R_i, ACL_i)_{i=1,\ldots,N})$ to the database DB, the latter encrypts DB by running $(EDB, sk_{\mathrm{DB}}) \xleftarrow{\$} \mathsf{DBSetup}(pk_{\mathrm{I}}, DB)$, and sends the encrypted database $EDB = (pk_{\mathrm{DB}}, ER_1, \ldots ER_N)$ to all users $\mathsf{U}_1, \ldots, \mathsf{U}_M$.

When $\mathcal{E}$ sends a message $(\texttt{issue}, c)$ to user U_j, U_j engages in an Issue protocol with I on common input pk_{I} and category c, with I using sk_{I} as its secret input, at the end of which U_j obtains the access credential $cred_{c_i}$. User U_j returns a bit b to the environment indicating whether the issue protocol succeeded ($b = 1$) or failed ($b = 0$).

When $\mathcal{E}$ sends a message $(\texttt{transfer}, \sigma)$ to user U_j, then U_j first checks whether it has the necessary credentials $\{cred_c\}_{c \in ACL_\sigma}$ to access record R_σ. If so, she engages in a $\mathsf{Transfer}$ protocol with DB on common input $pk_{\mathrm{I}}, pk_{\mathrm{DB}}$, on U_j's private input σ and the relevant credentials $\{cred_c\}_{c \in ACL_\sigma}$, and on DB's private input sk_{DB}, until U_j obtains the record R_σ, or $\perp$ indicating failure. If the transfer succeeded she returns R_σ to the environment; if it failed, or the user didn't have the appropriate credentials, she returns $\perp$ to the environment.

We note that I and DB do not return any outputs to the environment.

THE IDEAL WORLD. In the ideal world all participants communicate through a trusted party T which implements the functionality of our protocol. We describe the behavior of T on the inputs of the ideal-world users $\mathsf{U}'_1, \ldots, \mathsf{U}'_M$, the ideal-world issuer I', and the ideal-world database DB'.

The trusted party T maintains an initially empty set C_i for each user U'_i and sets $DB \leftarrow \perp$. It responds to queries from the different parties as follows.

- Upon receiving $(\texttt{initdb}, (R_i, ACL_i)_{i=1,\ldots,N})$ from DB', T sets $DB \leftarrow (R_i, ACL_i)_{i=1,\ldots,N}$.
- Upon receiving $(\texttt{issue}, c)$ from U'_i, T sends $(\texttt{issue}, \mathsf{U}'_i, c) = arg$ to I' who sends back a bit b. If $b = 1$ then the T adds c to C_i and sends b to U'_i; otherwise it simply sends b to U'_i.
- Upon receiving $(\texttt{transfer}, \sigma)$ from U'_i, T proceeds as follows. If $DB \neq \perp$, it sends $\texttt{transfer}$ to DB', who sends back a bit b. If $b = 1$ and $ACL_\sigma \subseteq C_i$, then it sends the record R_σ to U'_i. If $b = 0$ or $DB = \perp$ it sends $\perp$ to U'_i.

The ideal-world parties $\mathsf{U}'_1, \ldots, \mathsf{U}'_M, \mathsf{I}, \mathsf{DB}$ simply relay inputs and outputs between the environment $\mathcal{E}$ and the trusted party T.

SECURITY PROPERTIES. Let us discuss some of the security properties that the ideal world (and therefore also any secure real-world implementation) offers to the parties. It is easy to verify that these properties hold for the ideal world.

User Privacy: The database cannot tell which user makes a query, nor can it tell which record is being accessed. That is, the database only gets to know that some user accesses some record for which the user priorly obtained the necessary credentials. If the database colludes with the issuer and potentially with other users, then they can only try to identify the user or her selection based on which credentials were issued to whom, and which credentials are necessary to successfully access which record.

Database Security: A cheating user alone cannot access a record for which she does not have the necessary credentials. Colluding users cannot pool their credentials, meaning that they cannot access any records that none of them would have been able to obtain individually. If the issuer colludes with one or more users, they can only obtain as many records from the database as the number of transfer queries that were performed.

3 Preliminaries

Let $\mathsf{Pg}(1^\kappa)$ be a pairing group generator that on input 1^κ outputs descriptions of multiplicative groups $\mathbb{G}_1, \mathbb{G}_\mathrm{T}$ of prime order p where $|p| > \kappa$. Let $\mathsf{Pg}(p)$ be a pairing group generator that on input p outputs descriptions of multiplicative groups $\mathbb{G}_1, \mathbb{G}_\mathrm{T}$ of prime order p.

Let $\mathbb{G}_1^* = \mathbb{G}_1 \setminus \{1\}$ and let $g \in \mathbb{G}_1^*$. The generated groups are such that there exists an admissible bilinear map $e : \mathbb{G}_1 \times \mathbb{G}_1 \to \mathbb{G}_\mathrm{T}$, meaning that (1) for all $a, b \in \mathbb{Z}_p$ it holds that $e(g^a, g^b) = e(g, g)^{ab}$; (2) $e(g, g) \neq 1$; and (3) the bilinear map is efficiently computable.

Definition 3.1 We say that the ℓ-strong Diffie-Hellman (ℓ-SDH) assumption [4] holds in group $\mathbb{G}_1$ of order $p > 2^\kappa$ if for all polynomial-time adversaries A the advantage

$$\mathbf{Adv}^{\ell\mathrm{SDH}}_{\mathbb{G}_1}(\kappa) \;=\; \Pr\left[\,\mathsf{A}(g, g^x, \ldots, g^{x^\ell}) = (c, g^{1/(x+c)})\,\right]$$

is a negligible function in κ, where $g \xleftarrow{\$} \mathbb{G}_1^*$ and $x, c \xleftarrow{\$} \mathbb{Z}_p$.

Definition 3.2 We say that the decision ℓ-bilinear Diffie-Hellman exponent (ℓ-BDHE) assumption [6] holds in groups $\mathbb{G}_1, \mathbb{G}_\mathrm{T}$ of order $p > 2^\kappa$ if for all polynomial-time adversaries A the advantage $\mathbf{Adv}^{\ell\mathrm{BDHE}}_{\mathbb{G}_1,\mathbb{G}_\mathrm{T}}(\kappa)$ given by

$$\Pr\left[\,\mathsf{A}(g, h, g^\alpha, \ldots, g^{\alpha^{\ell-1}}, g^{\alpha^{\ell+1}}, \ldots, g^{\alpha^{2\ell}}, e(g,h)^{\alpha^\ell}) = 1\,\right] \\ -\; \Pr\left[\,\mathsf{A}(g, h, g^\alpha, \ldots, g^{\alpha^{\ell-1}}, g^{\alpha^{\ell+1}}, \ldots, g^{\alpha^{2\ell}}, S) = 1\,\right]$$

is a negligible function in κ, where $g, h \xleftarrow{\$} \mathbb{G}_1^*$, $S \xleftarrow{\$} \mathbb{G}_\mathrm{T}^*$ and $\alpha \xleftarrow{\$} \mathbb{Z}_p$

Definition 3.3 We say that the ℓ-power decision Diffie-Hellman (ℓ-PDDH) assumption [14] holds in groups $\mathbb{G}_1, \mathbb{G}_\mathrm{T}$ if for all polynomial-time adversaries A the advantage $\mathbf{Adv}^{\ell\mathrm{PDDH}}_{\mathbb{G}_1,\mathbb{G}_\mathrm{T}}(\kappa)$ given by

$$\Pr\left[\,\mathsf{A}(g, g^\alpha, \ldots, g^{\alpha^\ell}, H, H^\alpha, H^{\alpha^2} \ldots, H^{\alpha^\ell}) = 1\,\right] - \Pr\left[\,\mathsf{A}(g, g^\alpha, \ldots, g^{\alpha^\ell}, H, H_1, \ldots, H_\ell) = 1\,\right]$$

is a negligible function in κ, where $g \xleftarrow{\$} \mathbb{G}_1^*$, $H, H_1, \ldots, H_\ell \xleftarrow{\$} \mathbb{G}_\mathrm{T}^*$ and $\alpha \xleftarrow{\$} \mathbb{Z}_p$.

The ℓ-PDDH assumption is actually implied by the simpler ℓ-BDHE assumption, as kindly pointed out to us in personal communication by Brent Waters. For ease of presentation we use the PDDH assumption in security proofs; security under the BDHE assumption automatically follows.

Theorem 3.4 If the $(\ell+1)$-BDHE assumption holds in groups $\mathbb{G}_1, \mathbb{G}_\mathrm{T}$, then the ℓ-PDDH also holds in $\mathbb{G}_1, \mathbb{G}_\mathrm{T}$. More precisely,

$$\mathbf{Adv}^{\ell\mathrm{PDDH}}_{\mathbb{G}_1,\mathbb{G}_\mathrm{T}}(\kappa) \;\leq\; \ell \cdot \mathbf{Adv}^{(\ell+1)\mathrm{BDHE}}_{\mathbb{G}_1,\mathbb{G}_\mathrm{T}}(\kappa)\;.$$

Proof of (due to Brent Waters): The proof employs a hybrid argument. Consider the sequence of games **Game-i** for $i = 1, \ldots, \ell$ where an adversary A is given as input a tuple $(g, g^{\alpha}, g^{\alpha^2}, \ldots, g^{\alpha^{\ell}}, H, H^{\alpha}, H^{\alpha^2}, \ldots, H^{\alpha^i}, H_{i+1}, \ldots, H_{\ell})$, where $H, H_{i+1}, \ldots, H_{\ell} \stackrel{\$}{\leftarrow} \mathbb{G}_{\mathrm{T}}$ and the other inputs are as in the ℓ-PDDH game.

The ℓ-PDDH assumption says that it is hard to distinguish between **Game-0** and **Game-ℓ**. If algorithm A breaks the ℓ-PDDH assumption with advantage ϵ, then there must exist some $i \in \{0, \ldots, \ell\}$ such that A distinguishes **Game-i** from **Game-$(i+1)$** with probability ϵ/ℓ. Given this algorithm A, consider the following adversary B against the $(\ell+1)$-BDHE assumption.

On input $(g, h, g^{\alpha}, g^{\alpha^2}, \ldots, g^{\alpha^{\ell}}, g^{\alpha^{\ell+2}}, \ldots, g^{\alpha^{2\ell+2}}, S)$, algorithm B has to decide whether $S = e(g,h)^{\alpha^{\ell+1}}$ or random. It sets $H \leftarrow e(g^{\alpha^{\ell-i}}, h)$, $H_j \leftarrow e(g^{\alpha^{\ell-i+j}}, h)$ for $j = 1, \ldots, i$, and chooses $H_j \stackrel{\$}{\leftarrow} \mathbb{G}_{\mathrm{T}}$ for $j = i+2, \ldots, \ell$. It then runs A on input $(g, g^{\alpha}, g^{\alpha^2}, \ldots, g^{\alpha^{\ell}}, H, H_1, \ldots, H_i, S, H_{i+2}, \ldots, H_{\ell})$ to A.

It is clear that if $S = e(g,h)^{\alpha^{\ell+1}}$ then B perfectly simulates **Game-i**, while if S is random then it perfectly simulates **Game-$i+1$**. It can therefore win its own game with probability ϵ/ℓ simply outputting whatever A outputs. ∎

3.1 Modified Boneh-Boyen Signatures

We use the following modification of the weakly-secure signature scheme by Boneh and Boyen [4]. The scheme uses a pairing generator Pg as defined above.

The signer's secret key is $(x_m, x_1, \ldots, x_l) \stackrel{\$}{\leftarrow} \mathbb{Z}_p$, the corresponding public key is $(g, y_m = g^{x_m}, y_1 = g^{x_1}, \ldots, y_l = g^{x_l})$ where g is a random generator of $\mathbb{G}_1$. The signature on the tuple of messages $(m, c_1, \ldots, c_l)$ is the following $s \leftarrow g^{1/(x_m + m + x_1 c_1 + \ldots + x_l c_l)}$; verification is done by checking whether $e(s, y_m \cdot g^m \cdot y_1^{c_1} \cdot \ldots \cdot y_l^{c_l}) = e(g,g)$ is true.

Security against *weak* chosen-message attacks is defined through the following game. The adversary begins by outputting N tuples of messages $((m_1, c_{1,1}, \ldots, c_{1,l}), \ldots, (m_N, c_{N,1}, \ldots, c_{N,l}))$. The challenger then generates the key pair and gives the public key to the adversary, together with signatures $s_1, \ldots, s_N$ on the message tuples. The adversary wins if it succeeds in outputting a valid signature s on a tuple $(m, c_1, \ldots, c_l) \notin \{(m_1, c_{1,1}, \ldots, c_{1,l}), \ldots, (m_N, c_{N,1}, \ldots, c_{N,l})\}$.

The scheme is said to be unforgeable under weak chosen-message attack if no PPT adversary has non-negligible probability of winning this game. An adaptation of the proof of [4] can be used to show that this scheme is unforgeable under weak chosen-message attack if the $(N+1)$-SDH assumption holds. The proof is provided in Appendix **??** for completeness.

3.2 Zero-Knowledge Proofs and Σ-Protocols

We use various zero-knowledge proofs of knowledge [3, 24] protocols to prove knowledge of and statement about discrete logarithms such as (1) proof of knowledge of a discrete logarithm modulo a prime [34], (2) proof of knowledge of equality of (elements of) representations [21] , (3) proof that a commitment opens to the product of two other committed values [7, 13, 16], and also (4) proof of the disjunction or conjunction of any two of the previous [25].

When referring to the proofs above, we will follow the notation introduced by Camenisch and Stadler [15] and formally defined by Camenisch, Kiayias, and Yung [10]. For instance, $PK\{(a,b,c) : y = g^a h^b \wedge \tilde{y} = \tilde{g}^a \tilde{h}^c\}$ denotes a "*zero-knowledge Proof of Knowledge of integers a, b, c such that* $y = g^a h^b$ *and* $\tilde{y} = \tilde{g}^a \tilde{h}^c$ *holds*," where $y, g, h, \tilde{y}, \tilde{g}$, and $\tilde{h}$ are elements of some groups $G = \langle g \rangle = \langle h \rangle$

and $\tilde{G} = \langle \tilde{g} \rangle = \langle \tilde{h} \rangle$. The convention is that the letters in the parenthesis (a, b, c) denote quantities of which knowledge is being proven, while all other values are known to the verifier.

Given a protocol in this notation, it is straightforward to derive actual protocol implementing the proof. Indeed, the computational complexities of the proof protocol can be easily derived from this notation: basically for each term $y = g^a h^b$, the prover and the verifier have to perform an equivalent computation, and to transmit one group element and one response value for each exponent. We refer to, e.g., Camenisch, Kiayias, and Yung [10] for details on this.

3.3 Credential Signature Scheme

We use the signature scheme proposed and proved secure by Au et al. [2], which is based on the schemes of Camenisch and Lysyankaya [12] and of Boneh et al. [5].

It assumes cyclic groups $\mathbb{G}$ and $\mathbb{G}_T$ of order p and a bilinear map $e : \mathbb{G} \times \mathbb{G} \to \mathbb{G}_T$. The signer's secret key is a random element $x \xleftarrow{\$} \mathbb{Z}_q$. The public key contains a number of random bases $g_1, h_0, \ldots, h_\ell, h_{\ell+1} \xleftarrow{\$} \mathbb{G}$, where $\ell \in \mathbb{N}$ is a parameter, and $y \leftarrow g_1^x$.

A signature on messages $m_0, \ldots, m_\ell \in \mathbb{Z}_p$ is a tuple (A, r, s) where $r, s \xleftarrow{\$} \mathbb{Z}_p$ are values chosen at random by the signer and $A = (g_1 h_0^{m_0} \cdots h_\ell^{m_\ell} h_{\ell+1}^r)^{1/(x+s)}$. Such a signature can be verified by checking whether $e(A, g_1^s y) = e(g_1 h_0^{m_0} \cdots h_\ell^{m_\ell} h_{\ell+1}^r, g_1)$.

Now assume that we are given a signature (A, r, s) on messages $m_0 \ldots, m_\ell \in \mathbb{Z}_p$ and want to prove that we indeed possess such a signature. To this end, we need to augment the public key with values $u, v \in \mathbb{G}$ such that $\log_{g_1} u$ and $\log_{g_1} v$ are not known. This can be done by choosing random values $t, t' \xleftarrow{\$} \mathbb{Z}_p$, computing $\tilde{A} = Au^t$, $B = v^t u^{t'}$ and executing the following proof of knowledge

$$PK\{(\alpha, \beta, s, t, t', m_0, \ldots, m_\ell, r) : B = v^t u^{t'} \ \wedge \ 1 = B^{-s} v^\alpha u^\beta \ \wedge$$
$$\tfrac{e(\tilde{A},y)}{e(g_1,g_1)} = e(\tilde{A}, g_1)^{-s} e(u, y)^t e(u, g_1)^\alpha e(h_{\ell+1}, g_1)^r \textstyle\prod_{i=0}^{\ell} e(h_i, g_1)^{m_i}\},$$

where $\alpha = st$ and $\beta = st'$.

It was proved in [2] that the above signature is unforgeable under adaptively chosen message attack if q-SDH assumption holds, where q is the number of signature queries, and that the associated PoK is perfect honest-verifier zero-knowledge.

4 Our Construction

We now describe our scheme in detail. We model access control lists as tuples of exactly ℓ categories $ACL_i = (c_{i1}, \ldots, c_{i\ell}) \in \mathcal{C}^\ell$. A record can therefore be associated with at most ℓ categories; unused entries are filled with a dummy category $c_{ij} = \texttt{dummy}$ for which we assume every user is given a credential for free. To issue anonymous credentials we employ the signature scheme presented in Section 3.3. and to implement the oblivious access control we extend the protocol by Camenisch et al. [14]. We will also use a number of proof protocols about discrete logarithms as described in Section 3.2.

Initial Setup We now describe the setup procedures of the issuer and the database provider. Users do not have their own setup procedure.

$\underline{\mathsf{ISetup}(\mathcal{C})}$:

$(\mathbb{G}, \mathbb{G}_{\mathrm{T}}, p) \xleftarrow{\$} \mathsf{Pg}(1^{\kappa})$; $g_t, h_t \xleftarrow{\$} {\mathbb{G}_{\mathrm{T}}}^*$; $g_1, h_0, h_1, h_2, u, v \xleftarrow{\$} \mathbb{G}^*$
$x_I \xleftarrow{\$} \mathbb{Z}_p$; $y_I \leftarrow g_1^{x_I}$
$sk_{\mathrm{I}} \leftarrow x_I$; $pk_{\mathrm{I}} \leftarrow (g_1, h_0, h_1, h_2, u, v, w, g_t, h_t, y_I)$
Return $(sk_{\mathrm{I}}, pk_{\mathrm{I}})$

Figure 2: Issuer Setup algorithm

To set up its keys, the issuer runs the randomized ISetup algorithm displayed in Figure 2. This will generate groups of prime order p, a public key pk_{I} and corresponding secret key sk_{I} for security parameter κ and category universe $\mathcal{C}$. He publishes the public key as a system-wide parameter.

$\underline{\mathsf{DBSetup}\big(pk_{\mathrm{I}}, DB = (R_i, ACL_i)_{i=1,\ldots,N}\big)}$:

$(\overline{\mathbb{G}}, \overline{\mathbb{G}}_{\mathrm{T}}) \xleftarrow{\$} \mathsf{Pg}(p)$; $g, h \xleftarrow{\$} \overline{\mathbb{G}}^*$; $H \leftarrow \overline{e}(g, h)$
$x_{\mathrm{DB}} \xleftarrow{\$} \mathbb{Z}_p$; $y_{\mathrm{DB}} \leftarrow g^{x_{\mathrm{DB}}}$
For $i = 1, \ldots, \ell$ do
    $x_i \xleftarrow{\$} \mathbb{Z}_p$; $y_i \leftarrow g^{x_i}$
$sk_{\mathrm{DB}} \leftarrow (h, x_{\mathrm{DB}}, x_1, \ldots, x_\ell)$; $pk_{\mathrm{DB}} \leftarrow (g, H, y_{\mathrm{DB}}, y_1, \ldots, y_\ell)$
For $i = 1, \ldots, N$ do
    Parse ACL_i as $(c_{i1}, \ldots c_{i\ell})$
    $E_i \leftarrow g^{\frac{1}{x_{\mathrm{DB}}+i+\sum_{j=1}^{\ell} x_j \cdot c_{ij}}}$
    $F_i \leftarrow e(h, E_i) \cdot R_i$
    $ER_i \leftarrow (E_i, F_i)$
Return $\big((pk_{\mathrm{DB}}, ER_1, \ldots, ER_N),\ sk_{\mathrm{DB}}\big)$

Figure 3: Database Setup algorithm

To set up the database, the database provider runs the algorithm shown in Figure 3. That is, it uses the issuer's public key and a pairing group generator to create goups of the same order p and generate keys for encrypting records. First the database provider chooses its secret key x_{DB}. Next he encrypts each record R_i as (E_i, F_i), each with its own key. These keys not only depend on the database provider's secret key (x_{DB}), but also on the index of the record (i) and the categories defined in the access control policy for the record ($\{x_c\}_{c \in \bigcup_{i=1}^{N} ACL_i}$). The pairs (E_i, F_i) can be seen as an ElGamal encryption [29] in $\overline{\mathbb{G}}_{\mathrm{T}}$ of R_i under the public key H. But instead of using random elements from $\overline{\mathbb{G}}_{\mathrm{T}}$ as the first component, our protocol uses verifiably random [28] values $E_i = g^{\frac{1}{x_{\mathrm{DB}}+i+\sum_{j=1}^{\ell} x_j \cdot c_{ij}}}$. It is this verifiability that during the transfer phase allows the database to check that the user is indeed asking for the decryption key for one particular records with a particular access control policy for which user has appropriate credentials.

Issuing Credentials To be able to make database queries, a user needs to obtain the credentials for the categories she is allowed to access. To this end, the user runs the Issue protocol with the issuer as depicted in Figure 4. We leave open how the issuer determines which user has access to which categories, but we do assume that the communication links are authenticated so that the issuer knows which user it is talking to.

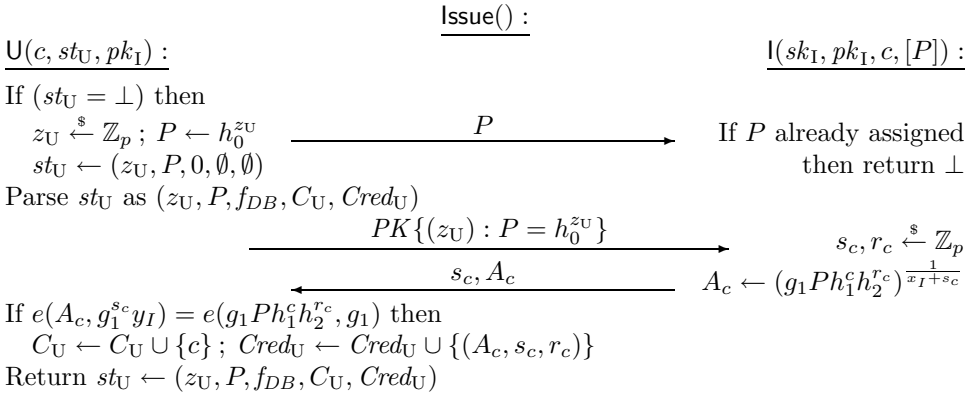

Figure 4: Issue protocol

Apart from the issuer's public key, the user's input also includes her state $st_{\mathrm{U}} = (z_{\mathrm{U}}, P, f_{DB}, C_{\mathrm{U}}, Cred_{\mathrm{U}})$, which is a tuple containing her master secret, her pseudonym, a bit f_{DB} indicating whether she already accessed the database, the set of categories C_{U} to which she currently has access, and the corresponding credentials $Cred_{\mathrm{U}}$. The input of the issuer contains his secret and public key, the category c for which the user wants a credential, and the pseudonym P of the user, if she registered one before.

If the user runs the issuing protocol for the first time, her input will contain the empty state ($st_{\mathrm{U}} = \perp$). In this case, the user first generates her master secret z_{U} and calculates her pseudonym $P = h_0^{z_{\mathrm{U}}}$, sends P to the issuer, and then initializes her state as $st_{\mathrm{U}} = (z_{\mathrm{U}}, P, 0, \emptyset, \emptyset)$.

As a result of the issuing protocol, the user will obtain an access credential for the category $c \in \mathcal{C}$. This credential is a tuple $cred_c = (A_c, s_c, r_c)$ which can be verified by checking $e(A_c, g_1^{s_c} y_I) = e(g_1 P h_1^c h_2^{r_c}, g_1)$. We assume that the user and the issuer run the issuing protocol for each category for which the user is allowed to obtain a credential individually. It is not hard to see how to issue the credentials for all of the user's categories at once.

We note that credential (A_c, s_c, r_c) is a signature as defined in Section 3.3 on the set of messages (z_{U}, c), where z_{U} is the user's master secret.

Accessing a Record When the user wants to access a record in the database, she engages in a Transfer protocol (Figure 5) with the database server.

The input of the database server is her secret and public key as well as the public key of the issuer. The input of the user is the index σ of the record that she wants to access, the encryption $ER_\sigma = (E_\sigma, F_\sigma)$ of that record, the access control policy of the records, her state (containing all her credential), and the public keys of the issuer and the database.

If this is the first transfer protocol she executes with this database (i.e., $f_{DB} = 0$), then the user asks the database to execute a proof of knowledge of the database secret key h. This zero-knowledge proof will enable to decrypt the contents of the database in the security proof.

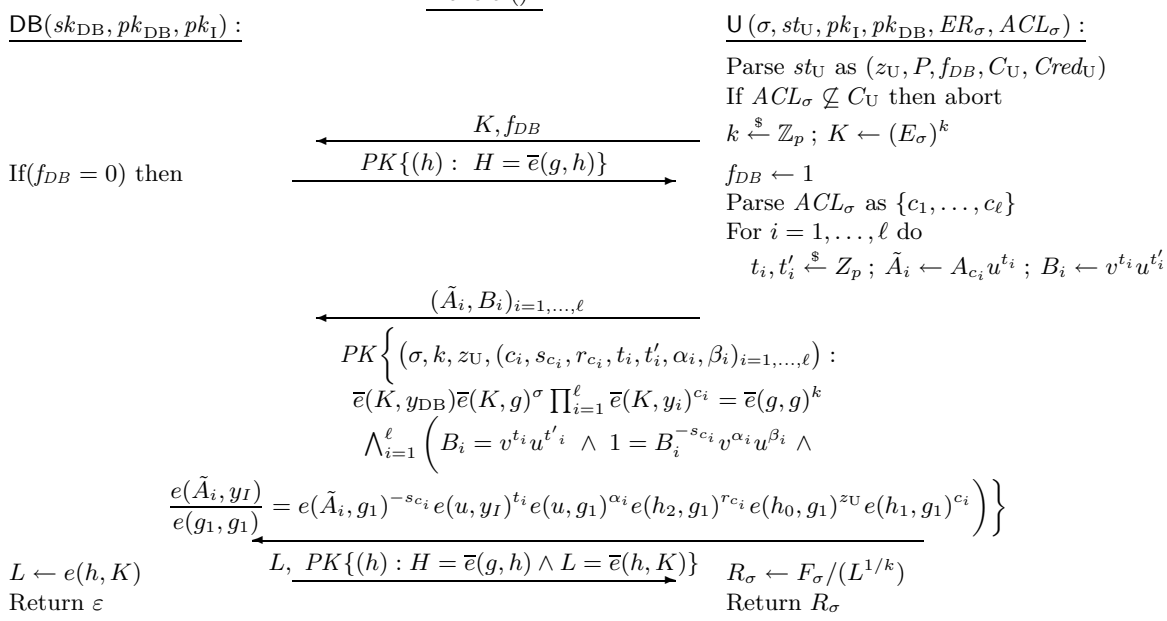

Figure 5: Transfer protocol

Then the user randomizes E_σ and sends this randomized version K to the database. Note that E_σ is derived from the database provider's secret key, the index of the records, and, most importantly all the categories of the record.

Next the user proves that K is correctly formed as a randomization of some E_i for which she possesses all necessary credentials. If the database provider accepts the proof, it computes L from h and K, sends L to the user, and proves that L was computed correctly. The protocol is easily seen to be correct by observing that $L = e(h, E_\sigma)^k$, so therefore $F_\sigma / L^{1/k} = R_\sigma$.

5 Security Analysis

The security of our protocol is analyzed by proving indistinguishability between adversary actions in the real protocol and in an ideal scenario that is secure by definition.

Given a real-world adversary A, we construct an ideal-world adversary A' such that no environment $\mathcal{E}$ can distinguish whether it is interacting with A or A'. We organize the proof in sublemmas according to which subset of parties are corrupted. We do not consider the cases where all parties are honest, where all parties are dishonest, where the issuer is the only honest party, or where the issuer is the only dishonest party, as these cases have no real practical interest.

For each case we prove the indistinguishability between the real and ideal worlds by defining a sequence of hybrid games **Game-0**, ..., **Game-n**. In each game we define a simulator Sim_i that runs A as a subroutine and that provides $\mathcal{E}$'s entire view. We define $\mathbf{Hybrid}_{\mathcal{E},\mathsf{Sim}_i}(\kappa)$ to be the probability that $\mathcal{E}$ outputs 1 when run in the world provided by Sim_i. The games are always constructed such that the first simulator Sim_0 runs A and all honest parties exactly like in the real world, so that $\mathbf{Hybrid}_{\mathcal{E},\mathsf{Sim}_0}(\kappa) = \mathbf{Real}_{\mathcal{E},\mathsf{A}}(\kappa)$, and such that the final simulator Sim_n is easily transformed into an ideal-world adversary A' so that $\mathbf{Hybrid}_{\mathcal{E},\mathsf{Sim}_n}(\kappa) = \mathbf{Ideal}_{\mathcal{E},\mathsf{A}'}(\kappa)$. By upper-bounding and summing the mutual game distances $\mathbf{Hybrid}_{\mathcal{E},\mathsf{Sim}_i}(\kappa) - \mathbf{Hybrid}_{\mathcal{E},\mathsf{Sim}_{i+1}}(\kappa)$ for $i = 0, \ldots, n-1$, we obtain an upper bound for the overall distance $\mathbf{Real}_{\mathcal{E},\mathsf{A}}(\kappa) - \mathbf{Ideal}_{\mathcal{E},\mathsf{A}'}(\kappa)$.

Theorem 5.1 If the $(N+2)$-BDHE assumption holds in $\mathbb{G}_1, \mathbb{G}_T$ and the q-SDH assumption holds in $\mathbb{G}_1$, then the $\mathcal{AC}$-$\mathcal{OT}$ protocol depicted in Figures 1–4 securely implements the AC-OT functionality, where N is the number of database records, q_{I} is the number of issued credentials, and $q = \max(q_{\mathrm{I}}, N+1)$.

We prove the theorem by separately proving it for all relevant combinations of corrupted parties in the lemmas below.

Lemma 5.2 For all environments $\mathcal{E}$ and all real-world adversaries A controlling the issuer and the database there exists an ideal-world adversary A' such that

$$\mathbf{Real}_{\mathcal{E},\mathsf{A}}(\kappa) - \mathbf{Ideal}_{\mathcal{E},\mathsf{A}'}(\kappa) \;\leq\; 2^{-\kappa}$$

Proof: Since the adversary can always simulate additional users himself, we can simplify the setting to a single honest user U.

Game-1 : Simulator Sim_1, at the first transfer query dictated by $\mathcal{E}$, runs the extractor for the proof of knowledge $PK\{(h) : H = \overline{e}(g,h)\}$ to extract from A the element h such that $\overline{e}(g,h) = H$. If the extractor fails, then Sim_1 outputs $\perp$ to $\mathcal{E}$; otherwise, it continues to run A interacting with the honest user algorithm. The difference between the two games is given by the knowledge error of the proof of knowledge, i.e.,

$$\mathbf{Hybrid}_{\mathcal{E},\mathsf{Sim}_0}(\kappa) - \mathbf{Hybrid}_{\mathcal{E},\mathsf{Sim}_1}(\kappa) \;\leq\; 2^{-\kappa} \;.$$

Game-2 : Simulator Sim_2 runs exactly like Sim_1, except that during each transfer phase it lets the user algorithm query a record R_j chosen at random among those for which it has the necessary credentials, rather than querying σ_i as imposed by $\mathcal{E}$. We claim that

$$\mathbf{Hybrid}_{\mathcal{E},\mathsf{Sim}_1}(\kappa) = \mathbf{Hybrid}_{\mathcal{E},\mathsf{Sim}_2}(\kappa) \;.$$

The claim follows from the (perfect) zero-knowledgeness of the proof of knowledge of $(\sigma, k, z_{\mathrm{U}}, \ldots)$ [2].

We now construct, based on the real-world adversary A, an ideal-world adversary A' that plays the simultaneous roles of the issuer and the database, and that incorporates all steps from the last game. The adversary A' simply relays all messages between the environment $\mathcal{E}$ and A. A' runs A to obtain the issuer's public key pk_{I} and the encrypted database $EDB = (pk_{\mathrm{DB}}, (E_1, F_1), \ldots, (E_N, F_N))$. Upon receiving $(\texttt{issue}, \mathsf{U}', c)$ from T, it executes the user's side of the issue protocol with A, maintaining state as necessary. If the resulting credential is valid, A' returns $b = 1$ to T, otherwise it returns $b = 0$. The first time it receives a message `transfer` from T, A' extracts h from A in the first proof of knowledge, uses it to decrypt R_i as $F_i / e(h, E_i)$ for $i = 1, \ldots, N$ and sends $(\texttt{initdb}, R_i, ACL_i)_{i=1,\ldots,N}$ to T. It then simulates an honest user querying for record R_j chosen at random among those for which it has the necessary credentials. If the transfer succeeds, A' sends $b = 1$ back to T; if it fails, it sends back $b = 0$. Later `transfer` queries are treated the same way, but without the first step of decrypting the database.

One can see that A' provides A with exactly the same environment as Sim_2 did, so we have

$$\mathbf{Ideal}_{\mathcal{E},\mathsf{A}'}(\kappa) \;=\; \mathbf{Hybrid}_{\mathcal{E},\mathsf{Sim}_2}(\kappa) \;.$$

Summing up all the above equations yields the lemma statement. ∎

Lemma 5.3 For all environments $\mathcal{E}$ and all real-world adversaries A controlling only the database there exists an ideal-world adversary A' such that

$$\mathbf{Real}_{\mathcal{E},\mathsf{A}}(\kappa) - \mathbf{Ideal}_{\mathcal{E},\mathsf{A}'}(\kappa) \;\leq\; 2^{-\kappa}$$

Proof: Since the adversary can always simulate additional users himself, here we also consider a simplified model with a single honest user U.

Game-0 : Simulator Sim_0 runs the adversary A, the honest user U and the honest issuer I exactly as in the real world, based on the input queries dictated by $\mathcal{E}$, so that

$$\mathbf{Hybrid}_{\mathcal{E},\mathsf{Sim}_0}(\kappa) = \mathbf{Real}_{\mathcal{E},\mathsf{A}}(\kappa) \;.$$

Game-1 : Simulator Sim_1, at the first transfer query dictated by $\mathcal{E}$, runs the extractor for the proof of knowledge $PK\{(h) :\; H = \overline{e}(g,h)\}$ to extract from A the element h such that $\overline{e}(g,h) = H$. If the extractor fails, then Sim_1 outputs $\perp$ to $\mathcal{E}$; otherwise, it continues to run A interacting with the honest user algorithm. The difference between the two games is given by the knowledge error of the proof of knowledge, i.e.,

$$\mathbf{Hybrid}_{\mathcal{E},\mathsf{Sim}_0}(\kappa) - \mathbf{Hybrid}_{\mathcal{E},\mathsf{Sim}_1}(\kappa) \;\leq\; 2^{-\kappa} \;.$$

Game-2 : Simulator Sim_2 runs exactly like Sim_1, except that during each transfer phase it lets the user algorithm query a record R_j chosen at random among those for which it has the necessary credentials, rather than querying σ_i as imposed by $\mathcal{E}$. We claim that

$$\mathbf{Hybrid}_{\mathcal{E},\mathsf{Sim}_1}(\kappa) = \mathbf{Hybrid}_{\mathcal{E},\mathsf{Sim}_2}(\kappa) \;.$$

The claim follows directly from the zero-knowledgeness of the proof of knowledge of $(\sigma, k, z_{\mathrm{U}}, ..)$ which includes perfect zero-knowledge proof of possessing a credential signature.

We now construct, based on the real-world adversary A, an ideal-world adversary A' that plays only the role of the database, and that incorporates all steps from the last game. The adversary A' simply relays all messages between the environment $\mathcal{E}$ and A. A' runs A to obtain the issuer's public key pk_{I} and the encrypted database $EDB = (pk_{\mathrm{DB}}, (E_1, F_1), \ldots, (E_N, F_N))$. The first time it receives a message `transfer` from T, A' extracts h from A in the first proof of knowledge, uses it to decrypt R_i as $F_i/e(h, E_i)$ for $i = 1, \ldots, N$ and sends $(\texttt{initdb}, R_i, ACL_i)_{i=1,\ldots,N}$ to T. It then simulates an honest user querying for record R_j chosen at random among those for which it has the necessary credentials. If the transfer succeeds, A' sends $b = 1$ back to T; if it fails, it sends back $b = 0$. Later `transfer` queries are treated the same way, but without the first step of decrypting the database.

One can see that A' provides A with exactly the same environment as Sim_2 did, so we have

$$\mathbf{Ideal}_{\mathcal{E},\mathsf{A}'}(\kappa) \;=\; \mathbf{Hybrid}_{\mathcal{E},\mathsf{Sim}_2}(\kappa)\ .$$

Summing up all the above equations yields the lemma statement. ∎

Lemma 5.4 For all environments $\mathcal{E}$ and all real-world adversaries A controlling only some of the users, there exists an ideal-world adversary A' such that

$$\begin{aligned}\mathbf{Real}_{\mathcal{E},\mathsf{A}}(\kappa) - \mathbf{Ideal}_{\mathcal{E},\mathsf{A}'}(\kappa) \le 2^{-\kappa}\cdot q_{\mathrm{T}} + \;\mathbf{Adv}^{\mathrm{q_I\text{-}SDH}}_{\mathbb{G}_1}(\kappa)\\ +\;\mathbf{Adv}^{\mathrm{(N+1)SDH}}_{\mathbb{G}_1}(\kappa) + \;(N+1)\cdot\mathbf{Adv}^{\mathrm{(N+2)BDHE}}_{\mathbb{G}_1,\mathbb{G}_{\mathrm{T}}}(\kappa)\ ,\end{aligned}$$

where q_{T} is the total number of transfer queries, q_{I} the number of issue queries, and N the number of records in the database.

Proof: Since the AC-OT functionality prevents users from pooling their credentials, we have to consider multiple users here, some of which are corrupted, and some of which are honest.

Game-1 : Simulator Sim_1, at each transfer query by a corrupted user dictated by $\mathcal{E}$, runs the extractor for the proof of knowledge $PK\{(\sigma, k, z_{\mathrm{U}}, \ldots)\}$ to extract from A the witness $(\sigma', k, \{cred'_{c'}\}, \{c'\})$. If the extractor fails, then Sim_1 outputs $\perp$ to $\mathcal{E}$; otherwise, it continues to run A interacting with the honest database algorithm. The difference between the two games is given by t times the knowledge error of the proof of knowledge, i.e.,

$$\mathbf{Hybrid}_{\mathcal{E},\mathsf{Sim}_0}(\kappa) - \mathbf{Hybrid}_{\mathcal{E},\mathsf{Sim}_1}(\kappa) \;\le\; 2^{-\kappa}\cdot t\ .$$

Note that the time required to execute these t extractions is t times the time of doing a single extraction, because the transfer protocols can only run sequentially, rather than concurrently.

Game-2 - Simulator Sim_2 runs exactly like Sim_1, except that Sim_2 outputs $\perp$ to $\mathcal{E}$, if at least one of the extracted values $\{cred'_{c'}\}$ was not issued during any of the Issue protocols. One can see that in this case $A_{c'}$ is a forged credential signature on c'. Note that this also includes the case that corrupted users manage to pool their credentials. Since only a single value z_{U} is extracted, one of the pooled credentials must have a different z_{U} value than when it was issue, and hence must be a forged credential. The difference between **Game-1** and **Game-2** is bounded by the following claim:

Claim 5.5 We have that

$$\mathbf{Hybrid}_{\mathcal{E},\mathsf{Sim}_1}(\kappa) - \mathbf{Hybrid}_{\mathcal{E},\mathsf{Sim}_2}(\kappa) \;\le\; \mathbf{Adv}^{\mathrm{q_I\text{-}SDH}}_{\mathbb{G}_1}(\kappa)\ ,$$

It is obvious that if the extracted credentials were not legally issued then adversary A broke the credential signature scheme. By the security proof given in [2], this directly gives rise to an expected polynomial-time adversary with non-negligible advantage in solving the q_{I}-SDH problem, where q_{I} is the number issue queries made by the adversary.

Game-3 - Simulator Sim_3 runs exactly like Sim_2, except that Sim_3 outputs $\perp$ to $\mathcal{E}$, if the extracted number of the record $\sigma' \notin \{1, \ldots, N\}$ or the extracted set of categories $\{c'\}$ does not match ACL'_σ during any of the transfers. One can see that in this case $s = K^{1/k}$ is a forged modified Boneh-Boyen signature (described in section 3.3) on record $R_{\sigma'}$.

The difference between **Game-2** and **Game-3** is bounded by the following claim; we postpone the proof of the claim until later.

Claim 5.6 We have that

$$\mathbf{Hybrid}_{\mathcal{E},\mathsf{Sim}_2}(\kappa) - \mathbf{Hybrid}_{\mathcal{E},\mathsf{Sim}_3}(\kappa) \;\leq\; \mathbf{Adv}^{(\mathrm{N+1})\mathrm{SDH}}_{\mathbb{G}_1}(\kappa) \;.$$

Game-4 : Simulator Sim_4, at the first transfer query dictated by $\mathcal{E}$, runs the simulated proof of knowledge $PK\{(h) : \; H = \overline{e}(g, h)\}$ The value L returned in each transfer query is computed as $L \leftarrow (F_\sigma / R_\sigma)^k$, and the final PK in the transfer phase is replaced by a simulated proof.

Note that now the simulation of the transfer phase no longer requires knowledge of h. However, all of the simulated proofs are proofs of true statements and the change in the computation of L is purely conceptional. Thus by the perfect zero-knowledge property, we have that

$$\mathbf{Hybrid}_{\mathcal{E},\mathsf{Sim}_4}(\kappa) \;=\; \mathbf{Hybrid}_{\mathcal{E},\mathsf{Sim}_3}(\kappa) \;.$$

Game-5 : Simulator Sim_5 replaces the values $F_1, \ldots, F_N$ sent to A during $\mathsf{DBSetup}$ phase with random elements from $\mathbb{G}_\mathrm{T}$. Now at this point, the second proof in the previous game is a simulated proof of a false statement. Intuitively, if these changes enable an environment $\mathcal{E}$ to separate the experiments, then one can solve an instance of the BDHE problem. This is captured in the following claim, that we will prove later:

Claim 5.7 We have that

$$\mathbf{Hybrid}_{\mathcal{E},\mathsf{Sim}_4}(\kappa) - \mathbf{Hybrid}_{\mathcal{E},\mathsf{Sim}_5}(\kappa) \;\leq (N+1) \cdot \mathbf{Adv}^{(\mathrm{N+2})\mathrm{BDHE}}_{\mathbb{G}_1,\mathbb{G}_\mathrm{T}}(\kappa) \;.$$

We now construct, based on the real-world adversary A, an ideal-world adversary A' that plays the role of the cheating user and performs all of the changes to the experiments described in the previous games except that at the time of the transfer, after having extracted the value of σ from A, it queries credentials from the trusted third party T for all categories in ACL_σ. Next, A' queries T to obtain record R_σ. Then he uses this record to compute $L \leftarrow (F_\sigma / R_\sigma)^k$, and the final PK in the transfer phase is replaced by a simulated proof.

One can see that A' provides A with exactly the same environment as Sim_5 did, so we have

$$\mathbf{Ideal}_{\mathcal{E},\mathsf{A}'}(\kappa) \;=\; \mathbf{Hybrid}_{\mathcal{E},\mathsf{Sim}_5}(\kappa) \;.$$

The running time of A' is that of A plus that of $O(N^2)$ exponentiations, l extractions and l proof simulations, so is polynomial in the security parameter.

Summing up all the above equations yields the lemma statement. We have left to prove the claims used above.

Proof of of Claim 5.6: We prove the claim by constructing an adversary B that breaks the unforgeability under weak chosen-message attack of the modified Boneh-Boyen signature scheme. By the security proof given in Appendix **??**, this directly gives rise to an expected polynomial-time adversary with non-negligible advantage in solving the $(N+1)$-SDH problem.

Given an adversary A for that distinguishes between **Game-1** and **Game-2**, consider the forger B that outputs message tuples $(1, c_{1,1}, \ldots, c_{1,l}), \ldots, (N, c_{N,1}, \ldots, c_{N,\ell})$. When given public key $(y_{\mathrm{DB}}, y_1, \ldots, y_M)$ and signatures $E_1, \ldots, E_N$ it creates an encrypted database using these values $E_1, \ldots, E_N$ and a self-chosen value $h \xleftarrow{\$} \overline{\mathbb{G}}$. At the i-th transfer it extracts from A values $(\sigma, k, \{c_i\})$ such that

$$\overline{e}(K, y_{\mathrm{DB}})\overline{e}(K, g)^{\sigma} \prod_{j=1}^{\ell} \overline{e}(K, y_j)^{c_{i,j}} = \overline{e}(g, g)^k$$

(This extraction is guaranteed to succeed since we already eliminated failed extractions in the transition from **Game-0** to **Game-1**.) When $\sigma \notin \{1, \ldots, N\}$ or $c_{\sigma,j} \notin ACL_\sigma$ for some j then B outputs $s \leftarrow K^{1/k}$ as its forgery on message tuple $(\sigma, c_{\sigma,1}, \ldots, c_{\sigma,\ell})$. ■

Proof of of Claim 5.7: Given algorithms $\mathcal{E}, \mathsf{A}$ with non-negligible advantage in distinguishing **Game-4** and **Game-5**, consider the following algorithm B solving the $(N+1)$-PDDH problem. The claim follows by the reduction from the PDDH to the BDHE problem in Theorem 3.4.

On input $g, g^x, \ldots, g^{x^{N+1}}, H_0, H_1, \ldots, H_{N+1}$, B runs $\mathcal{E}$ and A as Sim_4 does until $\mathcal{E}$ instructs to create database $DB = ((R_1, ACL_1), \ldots, (R_N, ACL_N))$. At this point B chooses $x_1, \ldots, x_\ell \xleftarrow{\$} \mathbb{Z}_p$ and computes $d_i = i + x_1 c_{i,1} + \ldots + x_\ell c_{i,\ell}$ for $i = 1, \ldots, N$. Let $f(X) = \prod_{i=1}^{N}(X + d_i) = \sum_{i=0}^{N} \alpha_i X^i$ and let $f_i(X) = f(X)/(X + d_i) = \sum_{j=0}^{N-1} \beta_{i,j} X^j$ for $i = 1, \ldots, N$. B computes $g' \leftarrow g^{f(x)} = \prod_{i=0}^{N} \left(g^{x^i}\right)^{\alpha_i}$, $y_{\mathrm{DB}} \leftarrow g^{xf(x)} = \prod_{i=0}^{N} \left(g^{x^{i+1}}\right)^{\alpha_i}$, and $y_i \leftarrow g'^{x_i}$ for $i = 1, \ldots, \ell$. For $i = 1, \ldots, N$, it computes $E_i \leftarrow g'^{\frac{1}{x+d_i}} = \prod_{j=0}^{N-1} (g^{x^j})^{\beta_{i,j}}$ and $F_i \leftarrow \prod_{j=0}^{N-1} (H_j)^{\beta_{i,j}}$. Algorithm B feeds $\big(pk_{\mathrm{DB}} = (g', H_0, y_{\mathrm{DB}}, y_1, \ldots, y_\ell), (E_1, F_1), \ldots, (E_N, F_N)\big)$ as the encrypted database to A, and continues running $\mathcal{E}$ and A as under Sim_4. If $\mathcal{E}$ outputs a bit b, then B outputs the same bit b.

It is clear that if $H_i = H_0^{x^i}$ then the database is distributed exactly as in **Game-4**, while if $H_1, \ldots, H_N$ are random it is distributed exactly as in **Game-5**. The advantage of B in breaking the $(N+1)$-PDDH assumption is the same as $\mathcal{E}$'s advantage in distinguishing **Game-4** from **Game-5**.
■

■

Lemma 5.8 For all environments $\mathcal{E}$ and all real-world adversaries A controlling the issuer and one or more users, there exists an ideal-world adversary A' such that

$$\mathbf{Real}_{\mathcal{E},\mathsf{A}}(\kappa) - \mathbf{Ideal}_{\mathcal{E},\mathsf{A}'}(\kappa) \leq 2^{-\kappa} \cdot q + \\ + \mathbf{Adv}^{(\mathrm{N}+1)\mathrm{SDH}}_{\mathbb{G}_1}(\kappa) + (N+1) \cdot \mathbf{Adv}^{(\mathrm{N}+2)\mathrm{BDHE}}_{\mathbb{G}_1,\mathbb{G}_\mathrm{T}}(\kappa) ,$$

where q is the total number of transfer protocols, and N the number of records in the database.

Proof: Since the issuer is controlled by the adversary, the corrupted users can obtain all the credentials they want, so without loss of generality we can restrict the setting to a single corrupted user.

Game-0 : Simulator Sim_0 runs the adversary A and the honest database DB exactly as in the real world, based on the input queries dictated by $\mathcal{E}$ so that

$$\mathbf{Hybrid}_{\mathcal{E},\mathsf{Sim}_0}(\kappa) = \mathbf{Real}_{\mathcal{E},\mathsf{A}}(\kappa) \ .$$

Game-1 : Simulator Sim_1, at each transfer query dictated by $\mathcal{E}$, runs the extractor for the proof of knowledge $PK\{(\sigma, k, z_{\mathrm{U}}, \ldots)\}$ to extract from A the witness $(\sigma, k, \{cred_c\}, \{c\})$. If the extractor fails, then Sim_1 outputs $\perp$ to $\mathcal{E}$; otherwise, it continues to run A interacting with the honest database algorithm. The difference between the two games is given by q times the knowledge error of the proof of knowledge, i.e.,

$$\mathbf{Hybrid}_{\mathcal{E},\mathsf{Sim}_0}(\kappa) - \mathbf{Hybrid}_{\mathcal{E},\mathsf{Sim}_1}(\kappa) \ \leq \ 2^{-\kappa} \cdot q \ .$$

Note that the time required to execute these q extractions is q times the time of doing a single extraction, because the transfer protocols can only run sequentially, rather than concurrently.

Game-2 - Simulator Sim_2 runs exactly like Sim_1, except that Sim_2 outputs $\perp$ to $\mathcal{E}$, if the extracted value $\sigma \notin \{1, \ldots, N\}$ or the extracted set of categories $\{c\}$ does not match ACL_σ during any of the transfers. One can see that in this case $s = K^{1/k}$ is a forged modified Boneh-Boyen signature (described in section 3.3) on the record $R_{\sigma'}$. The difference between **Game-1** and **Game-2** is bounded by the following claim. The proof is identical to that of Claim 5.6 and hence omitted.

Claim 5.9 We have that

$$\mathbf{Hybrid}_{\mathcal{E},\mathsf{Sim}_2}(\kappa) - \mathbf{Hybrid}_{\mathcal{E},\mathsf{Sim}_1}(\kappa) \ \leq \ \mathbf{Adv}^{(\mathrm{N}+1)\mathrm{SDH}}_{\mathbb{G}_1}(\kappa) \ .$$

Game-3 : Simulator Sim_3, at the first transfer query dictated by $\mathcal{E}$, runs the simulated proof of knowledge $PK\{(h) : \ H = \overline{e}(g, h)\}$. The value L returned in each transfer query is computed as $L \leftarrow (F_\sigma / R_\sigma)^k$, and the final PK in the transfer phase is replaced by a simulated proof.

Note that now the simulation of the transfer phase no longer requires knowledge of h. However, all of the simulated proofs are the proofs of true statements and the change in the computation of L is purely conceptional. Thus by the perfect zero-knowledge property, we have

$$\mathbf{Hybrid}_{\mathcal{E},\mathsf{Sim}_3}(\kappa) \ = \ \mathbf{Hybrid}_{\mathcal{E},\mathsf{Sim}_2}(\kappa) \ .$$

Game-4 : Simulator Sim_4 replaces the values $F_1, \ldots, F_N$ sent to A during $\mathsf{DBSetup}$ phase with random elements from $\mathbb{G}_{\mathrm{T}}$. Now at this point, the second proof in the previous game is a simulated proof of a false statement. Intuitively, if these changes enable an environment $\mathcal{E}$ to separate the experiments, then one can solve the BDHE problem. This is captured in the following claim. The proof is identical to that of Claim 5.7 and therefore omitted.

Claim 5.10 It holds that

$$\mathbf{Hybrid}_{\mathcal{E},\mathsf{Sim}_4}(\kappa) = \mathbf{Hybrid}_{\mathcal{E},\mathsf{Sim}_3}(\kappa) \;\leq\; (N+1) \cdot \mathbf{Adv}^{(\mathrm{N}+2)\mathrm{BDHE}}_{\mathbb{G}_1,\mathbb{G}_\mathrm{T}}(\kappa) \;.$$

We now construct, based on the real-world adversary A, an ideal-world adversary A' that plays the simultaneous roles of the issuer and the user and performs all of the changes to the experiments described in the previous games except that at the time of the transfer, after having extracted the value of σ' from A, he queries T for the credentials for the all categories from ACL'_σ. As the adversary simultaneously plays the role of the issuer it sends bit 1 to T for granting all the necessary credentials. After that A' queries T the index σ' to obtain the record $R_{\sigma'}$. Then he uses this record to compute $L \leftarrow (F_{\sigma'}/R_{\sigma'})^k$, and the final PK in the transfer phase is replaced by a simulated proof.

One can see that A' provides A with exactly the same environment as Sim_4 did, so we have

$$\mathbf{Ideal}_{\mathcal{E},\mathsf{A}'}(\kappa) \;=\; \mathbf{Hybrid}_{\mathcal{E},\mathsf{Sim}_4}(\kappa) \;.$$

Summing up all the above equations yields the lemma statement. ∎

Acknowledgements

This work was supported in part by the European Community through the Seventh Framework Programme (FP7/2007-2013) project PrimeLife (grant agreement no. 216483) and through the ICT programme project ECRYPT II (contract ICT-2007-216676).

References

[1] W. Aiello, Y. Ishai, O. Reingold. Priced oblivious transfer: How to sell digital goods. In *EUROCRYPT 2001*, *LNCS* vol. 2045, 119–135. Springer, 2001. (Cited on page 3.)

[2] M. H. Au, W. Susilo, Y. Mu. Constant-size dynamic k-TAA. In *SCN 06*, *LNCS* vol. 4116, 111–125. Springer, 2006. (Cited on pages 10, 15 and 17.)

[3] M. Bellare, O. Goldreich. On defining proofs of knowledge. In *CRYPTO '92*, *LNCS* vol. 740, 390–420. Springer, 1993. (Cited on page 9.)

[4] D. Boneh, X. Boyen. Short signatures without random oracles. In *EUROCRYPT 2004*, *LNCS* vol. 3027, 56–73. Springer, 2004. (Cited on pages 8 and 9.)

[5] D. Boneh, X. Boyen, H. Shacham. Short group signatures. In *CRYPTO 2004*, *LNCS* vol. 3152, 41–55. Springer, 2004. (Cited on page 10.)

[6] D. Boneh, C. Gentry, B. Waters. Collusion resistant broadcast encryption with short ciphertexts and private keys. In *CRYPTO 2005*, *LNCS* vol. 3621, 258–275. Springer, 2005. (Cited on page 8.)

[7] S. Brands. Rapid demonstration of linear relations connected by boolean operators. In *EUROCRYPT '97*, *LNCS* vol. 1233, 318–333. Springer, 1997. (Cited on page 9.)

[8] S. Brands. *Rethinking Public Key Infrastructure and Digital Certificates— Building in Privacy*. Ph.D. thesis, Eindhoven Institute of Technology, Eindhoven, The Netherlands, 1999. (Cited on page 2.)

[9] J. Camenisch, M. Dubovitskaya, G. Neven. Oblivious transfer with access control. Cryptology ePrint Archive, 2009. (Not cited.)

[10] J. Camenisch, A. Kiayias, M. Yung. On the portability of generalized Schnorr proofs. In *EUROCRYPT 2009*, *LNCS* vol. 5479, 425–442. Springer, 2009. (Cited on pages 9 and 10.)

[11] J. Camenisch, A. Lysyanskaya. Efficient non-transferable anonymous multi-show credential system with optional anonymity revocation. In *EUROCRYPT 2001*, *LNCS* vol. 2045, 93–118. Springer, 2001. (Cited on page 2.)

[12] J. Camenisch, A. Lysyanskaya. Signature schemes and anonymous credentials from bilinear maps. In *CRYPTO 2004*, *LNCS* vol. 3152, 56–72. Springer, 2004. (Cited on page 10.)

[13] J. Camenisch, M. Michels. Proving in zero-knowledge that a number n is the product of two safe primes. In *EUROCRYPT '99*, *LNCS* vol. 1592, 107–122. Springer, 1999. (Cited on page 9.)

[14] J. Camenisch, G. Neven, abhi shelat. Simulatable adaptive oblivious transfer. In *EUROCRYPT 2007*, *LNCS* vol. 4515, 573–590. Springer, 2007. (Cited on pages 2, 8 and 10.)

[15] J. Camenisch, M. Stadler. Efficient group signature schemes for large groups. In *CRYPTO '97*, *LNCS* vol. 1296, 410–424. Springer, 1997. (Cited on page 9.)

[16] J. L. Camenisch. *Group Signature Schemes and Payment Systems Based on the Discrete Logarithm Problem*. Ph.D. thesis, ETH Zürich, 1998. Diss. ETH No. 12520, Hartung Gorre Verlag, Konstanz. (Cited on page 9.)

[17] R. Canetti. *Studies in Secure Multiparty Computation and Applications*. Ph.D. thesis, Weizmann Institute of Science, Rehovot 76100, Israel, 1995. (Cited on page 6.)

[18] R. Canetti. Security and composition of multi-party cryptographic protocols. *Journal of Cryptology*, 13(1):143–202, 2000. (Cited on page 6.)

[19] D. Chaum. Security without identification: transaction systems to make big brother obsolete. *Communications of the ACM*, 28(10):1030–1044, 1985. (Cited on page 2.)

[20] D. Chaum, J.-H. Evertse. A secure and privacy-protecting protocol for transmitting personal information between organizations. In *CRYPTO '86*, *LNCS* vol. 263, 118–167. Springer, 1987. (Cited on page 2.)

[21] D. Chaum, T. P. Pedersen. Wallet databases with observers. In *CRYPTO '92*, *LNCS* vol. 740, 89–105. Springer-Verlag, 1993. (Cited on page 9.)

[22] L. Chen. Access with pseudonyms. In *Proceedings of the International Conference on Cryptography: Policy and Algorithms*, 232–243. Springer, 1995. (Cited on page 2.)

[23] S. Coull, M. Green, S. Hohenberger. Controlling access to an oblivious database using stateful anonymous credentials. Cryptology ePrint Archive, Report 2008/474, 2008. (Cited on pages 3 and 4.)

[24] R. Cramer, I. Damgård, P. D. MacKenzie. Efficient zero-knowledge proofs of knowledge without intractability assumptions. In *PKC 2000*, *LNCS* vol. 1751, 354–372. Springer, 2000. (Cited on page 9.)

[25] R. Cramer, I. Damgård, B. Schoenmakers. Proofs of partial knowledge and simplified design of witness hiding protocols. In *CRYPTO '94*, *LNCS* vol. 839, 174–187. Springer, 1994. (Cited on page 9.)

[26] I. Damgård. Payment systems and credential mechanisms with provable security against abuse by individuals. In *CRYPTO '88*, *LNCS* vol. 403, 328–335. Springer, 1990. (Cited on page 2.)

[27] G. Di Crescenzo, R. Ostrovsky, S. Rajagopalan. Conditional oblivious transfer and timed-release encryption. In *EUROCRYPT '99*, *LNCS* vol. 1592, 74–89. Springer, 1999. (Cited on page 3.)

[28] Y. Dodis, A. Yampolskiy. A verifiable random function with short proofs and keys. In *Public Key Cryptography – PKC 2005*, *LNCS* vol. 3386, 416–431. Springer, 2005. (Cited on page 11.)

[29] T. ElGamal. A public key cryptosystem and a signature scheme based on discrete logarithms. *IEEE Transactions on Information Theory*, 31(4):469–472, 1985. (Cited on page 11.)

[30] J. Herranz. Restricted adaptive oblivious transfer. Cryptology ePrint Archive, Report 2008/182, 2008. (Cited on page 3.)

[31] A. Lysyanskaya, R. Rivest, A. Sahai, S. Wolf. Pseudonym systems. In *Selected Areas in Cryptography*, *LNCS* vol. 1758. Springer, 1999. (Cited on page 2.)

[32] B. Pfitzmann, M. Waidner. Composition and integrity preservation of secure reactive systems. In *Proc. 7th ACM Conference on Computer and Communications Security*, 245–254. ACM press, 2000. (Cited on page 6.)

[33] B. Pfitzmann, M. Waidner. A model for asynchronous reactive systems and its application to secure message transmission. In *Proceedings of the IEEE Symposium on Research in Security and Privacy*, 184–200. IEEE Computer Society, IEEE Computer Society Press, 2001. (Cited on page 6.)

[34] C. P. Schnorr. Efficient signature generation for smart cards. *Journal of Cryptology*, 4(3):239–252, 1991. (Cited on page 9.)