Oblivious Transfer with Access Control : Realizing Disjunction without Duplication

Ye Zhang[1], Man Ho Au[2], Duncan S. Wong[3], Qiong Huang[3], Nikos Mamoulis[1], David W. Cheung[1], and Siu-Ming Yiu[1]

[1] Department of Computer Science, University of Hong Kong, Hong Kong
{yzhang4,nikos,dcheung,smyiu}@cs.hku.hk

[2] School of Computer Science and Software Engineering, University of Wollongong, Australia
aau@uow.edu.au

[3] Department of Computer Science, City University of Hong Kong, Hong Kong
duncan@cityu.edu.hk, csqhuang@gmail.com

Abstract. Oblivious Transfer with Access Control (AC-OT) is a protocol which allows a user to obtain a database record with a credential satisfying the access policy of the record while the database server learns nothing about the record or the credential. The only AC-OT construction that supports policy in *disjunctive form* requires duplication of records in the database, each with a different *conjunction* of attributes (representing one possible criterion for accessing the record). In this paper, we propose a new AC-OT construction secure in the standard model. It supports policy in disjunctive form directly, without the above duplication issue. Due to the duplication issue in the previous construction, the size of an encrypted record is in $O(\prod_{i=1}^{t} n_i)$ for a CNF policy $(A_{1,1} \vee \ldots \vee A_{1,n_1}) \wedge \ldots \wedge (A_{t,1} \vee \ldots \vee A_{t,n_t})$ and in $O(\binom{n}{k})$ for a k-of-n threshold gate. In our construction, the encrypted record size can be reduced to $O(\sum_{i=1}^{t} n_i)$ for CNF form and $O(n)$ for threshold case.

1 Introduction

When a user tries to obtain a record from a database, a conventional database server knows which record is being accessed and whether the user has access right to the record. However, for some applications, user privacy is a concern. In an outsourced medical database (e.g., Google Health [7]), knowing which records a user has accessed may leak private information about the user's medical condition to the service provider (i.e., Google). Also, knowing the user's access rights may provide hints to the service provider on what records the user may want to access. If the user has access right to diabetes related records, it is very likely that the user may have related medical issues. *Oblivious Transfer with Access Control (AC-OT)* [4] is a protocol designed for providing solutions to these user privacy problems.

In an AC-OT protocol, there is a database server, an issuer and a set of users. The issuer issues credentials to users where credentials are attributes which specify the access rights of the users. The database server has a database of records, each record is encrypted by the database server under a record-specific access policy $\mathbb{A}$. This encrypted database is accessible to all the users, e.g., by posting it onto a public web site. A user with credentials S (i.e. an attribute set) can obtain a record R anonymously by running the AC-OT transfer protocol with the database server if S satisfies the access policy $\mathbb{A}$ of R. The database server learns nothing except the fact that the user with proper credentials has obtained a record.

The access policy supported in the previous AC-OT construction [4] is for the conjunction of attributes (e.g., Student $\wedge$ CS Dept.). To realize disjunctive policy, same records may need to be duplicated and appear for multiple times in the database. Suppose the policy of a record R is $(A_1 \wedge A_2 \wedge A_3) \vee (B_1 \wedge B_2 \wedge B_3)$. R needs to be duplicated so that one encrypted R associates with a conjunctive policy $A_1 \wedge A_2 \wedge A_3$ and the other one with a conjunctive policy $B_1 \wedge B_2 \wedge B_3$. For a CNF policy $(A_{1,1} \vee \ldots \vee A_{1,n_1}) \wedge \ldots \wedge (A_{t,1} \vee \ldots \vee A_{t,n_t})$, the scheme in [4] produces a set of encrypted duplicated records of size $O(\prod_{i=1}^{t} n_i)$. For a k-of-n threshold policy, requiring at least k attributes in the set $\{A_1, \ldots, A_n\}$, the scheme produces a set of size $O(\binom{n}{k})$.

Our Result: In this paper, we propose a new AC-OT protocol and show that it is secure in the standard model. The construction supports policy in disjunctive form directly, without duplicating records. For policies expressed in CNF or threshold gate, the construction produces a smaller size of encrypted database. Specifically, the size is in $O(\sum_{i=1}^{t} n_i)$ for CNF type and in $O(n)$ for k-of-n threshold type of policies. Our construction idea is to use a signature in an oblivious transfer protocol which is then integrated with a ciphertext-policy attribute-based encryption (CP-ABE) scheme for supporting policies in the form of CNF and k-of-n threshold.

1.1 Related Work

In [6], Coull et al. considers AC-OT in a stateful environment, in which, the access policy (e.g., Biba and Bell-LaPadual) is represented by a graph. Each node in the graph is a state and each edge models a transaction from one state to another. Each user in a stateful environment is also assigned an initial state via a stateful anonymous credential. By using zero-knowledge proof, a user proves to the database server that he/she is in possession of a state and tries to access a record where the record corresponds to an edge from the user's current state. After the user obtains the record, the database server updates the user's credential to a new state. However, the scheme will be less efficient if it is used in a stateless environment which is considered in [4] and this paper. According to [4], an access policy in a stateless environment can be represented as a graph with nodes for each subset of attributes that a user could have access to, and with a self-loop edge for each record that can be accessed using this subset. If there are C attributes and N records, the graph will have 2^C nodes and up to N self-loop edges for each node, and the encrypted database will be of size $O(2^C N)$. Also, users have to update their credentials after each access, since it is fundamentally stateful.

The AC-OT construction proposed in this paper relies on a fully-simulatable adaptive oblivious transfer (OT) protocol and a ciphertext-policy attribute-based encryption (CP-ABE). A fully-simulatable adaptive OT is an adaptive OT in which the security is defined in a simulation based model (i.e., following an ideal-world/real-world paradigm). The first fully-simulatable adaptive OT was proposed by Camenisch et al. [5]. Two other protocols are due to Green and Hohenberger [8] and Jarecki and Liu [10]. In our construction, we employ the OT of [5] due to the special use of an unforgeable signature in the OT.

In [2], Bethencourt et al. proposed the first CP-ABE. It supports monotonic access structures and the security is proven in a selective model, where an adversary submits the challenge access policy $\mathbb{A}^*$ before obtaining the public key of CP-ABE. The CP-ABE used in our construction requires full security. Lewko et al. in [11] proposed the first fully secure CP-ABE in the standard model. The scheme supports access policies with linear secret sharing scheme (LSSS) [11,1].

2 Definitions and Security Model

2.1 Syntax

Let $k \in \mathbb{N}$ be a security parameter. Let $[a, b]$ be the set $\{i \in \mathbb{N} | a \leq i \leq b\}$ where $a, b \in \mathbb{Z}$. Let $y \overset{\$}{\leftarrow} A(x)$ be the assignment of y to the output of a probabilistic algorithm A on input x and a fresh random tape. We say that a function $f(k)$ is negligible in k if for all polynomial $p(k)$, there exists k' such that $f(k) < \frac{1}{p(k)}$ when all $k > k'$. Without loss of generality, we define a universe of attributes $\mathcal{U} = \{1, \ldots, |\mathcal{U}|\}$ and denote each attribute as an element of $\mathcal{U}$. Therefore, an attribute set $S \subseteq \mathcal{U}$. We also define an access policy $\mathbb{A}$ as a collection of attribute sets, i.e. $\mathbb{A} \subseteq 2^{\mathcal{U}} \backslash \{\}$.

An Oblivious Transfer with Access Control (AC-OT) protocol is a tuple of probabilistic polynomial-time (PPT) algorithms/protocols:

$ISetup(1^k)$: This issuer setup algorithm generates a public/secret key pair (pk_I, sk_I). The issuer runs the algorithm and publishes pk_I.

$DBSetup(pk_I, DB = (R_i, AP_i)_{i=1,\ldots,N})$: In a database DB with $N \in \mathbb{N}$ records (where N is another security parameter), $R_i \in \{0,1\}^*$ is the i-th record and AP_i is the access policy of R_i. This database setup algorithm generates a public/secret key pair (pk_{DB}, sk_{DB}) and encrypts R_i to encrypted records C_i with respect to AP_i. The database server runs this algorithm and publishes $(ER_i)_{i=1,\ldots,N} = (C_i, AP_i)_{i=1,\ldots,N}$ along with its public key pk_{DB}.

$Issue$: A user U engages in this protocol with the issuer. The inputs of U are an attribute set $S \subseteq \mathcal{U}$ and pk_I; and the input of the issuer is (pk_I, sk_I). By executing this protocol, U will obtain a credential $Cred_S \in \{0,1\}^*$ corresponding to S. We assume that the issuer has already authenticated U with respect to S.

$Transfer$: U engages in this protocol with the database server. The inputs of U are an index $\sigma \in [1, N]$, ER_σ, $Cred_S$ (w.r.t. S), pk_I and pk_{DB} where S satisfies the access policy AP_σ of ER_σ. The input of the database server is

(pk_{DB}, sk_{DB}). By executing this protocol, U will obtain R_σ if the execution is successful, otherwise, U will get $\perp$ (an error signal).

2.2 Security Model

The security of AC-OT is defined in a simulation-based model. In the model, there is a real world and an ideal world. In the real world, all parties communicate using a real AC-OT protocol π. Some of the parties are corrupted and controlled by an adversary $\mathcal{A}$. We call these parties as "dishonest parties". Other parties follow π honestly and are called "honest parties". In the ideal world, all honest parties and an adversary $\mathcal{A}'$ communicate by sending their outputs to, and receiving inputs from, a party T, which cannot be corrupted.

We say that π *securely implements* the functionality T if for any real-world adversary $\mathcal{A}$, there exists an ideal-world adversary $\mathcal{A}'$ such that no PPT distinguisher (the environment) Z can tell whether it interacts with $\mathcal{A}$ in the real world or with $\mathcal{A}'$ in the ideal world, with non-negligible probability. The environment Z provides inputs to all parties and interacts with the adversary arbitrarily.

Real world: Now we define how honest parties in the real world follow the AC-OT protocol π. The database server and issuer do not return anything to Z; only the users return to Z.

1. The issuer I runs $ISetup(1^k)$ to generate (pk_I, sk_I) and publishes pk_I.
2. Upon receiving $(issue, S)$ from environment Z, where $S \subseteq \mathcal{U}$, an honest user U engages in the Issue protocol with I. After running the protocol, U sends a bit b to Z indicating whether the protocol run is successful ($b = 1$) or not ($b = 0$). Note that if $b = 1$, U has obtained a credential $Cred_S$ for S.
3. When receiving $(initDB, DB = (R_i, AP_i)_{i=1,\ldots,N})$ from environment Z, the database server runs $DBSetup(pk_I, DB)$ to generate (pk_{DB}, sk_{DB}) and then creates encrypted records $(C_i)_{i=1,\ldots,N}$. It publishes $\{ER_i = (C_i, AP_i)\}_{i=1,\ldots,N}$ and pk_{DB}.
4. Upon receiving $(transfer, \sigma)$ from environment Z, U checks whether it has a credential corresponding to an attribute set satisfying AP_σ. If so, U engages in the Transfer protocol with the database server. To this end, U obtains R_σ if the protocol run is successful; otherwise, U receives an error signal $\perp$ from the database server. U also sends R_σ or $\perp$ to environment Z.

Ideal world: All parties communicate with each other via T. When receiving $(issue, \ldots)$, $(initDB, \ldots)$ or $(transfer, \ldots)$ from environment Z, honest parties forward it to T, then forward the outputs of T to Z. We now define the behavior of T. T maintains an attribute set S_U which is initially empty for each user U. T also sets $DB = \perp$.

1. Upon receiving $(issue, S)$ from U, T sends $(issue, U, S)$ to issuer I who, in turn, sends back a bit b. If $b = 1$, T sets $S_U = S$. Otherwise, T does nothing.
2. Upon receiving $(initDB, (R_i, AP_i)_{i=1,\ldots,N})$ from the database server, T sets $DB = (R_i, AP_i)_{i=1,\ldots,N}$.

3. Upon receiving $(transfer, \sigma)$ from U, T checks whether $DB = \perp$. If $DB \neq \perp$, it sends $(transfer)$ to the database server. If the database server sends back $b = 1$, T checks if $\sigma \in [1, N]$ and S_U satisfies the access policy AP_σ. If so, T sends R_σ to U. Otherwise, it sends $\perp$ to U.

3 Preliminaries

3.1 Bilinear Maps and Assumptions

Let $\mathbb{G}$ and $\mathbb{G}_T$ be two cyclic multiplicative groups of order n (which can be prime or composite). A bilinear map is defined as $\hat{e} : \mathbb{G} \times \mathbb{G} \rightarrow \mathbb{G}_T$ with the following properties: (1) Bilinear: for all $u, v \in \mathbb{G}$ and $a, b \in \mathbb{Z}$, $\hat{e}(u^a, v^b) = \hat{e}(u, v)^{ab}$; (2) Non-degenerate: if g is a generator of $\mathbb{G}$, then $\hat{e}(g, g) \neq 1_{\mathbb{G}_T}$; and (3) Computable: there exists an efficient algorithm to compute $\hat{e}(u, v)$ for any $u, v \in \mathbb{G}$.

Definition 1. (ℓ-strong Diffie-Hellman (ℓ-SDH) assumption) *This assumption holds in $\mathbb{G}$ if for all PPT adversaries $\mathcal{A}$, the advantage $\boldsymbol{Adv}_{\mathbb{G}}^{\ell\text{-}SDH}(k)$:*

$$\boldsymbol{Adv}_{\mathbb{G}}^{\ell\text{-}SDH}(k) = \Pr[\mathcal{A}(g, g^x, \ldots, g^{x^\ell}) = (c, g^{1/(x+c)})]$$

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

Definition 2. (ℓ-power Decisional Diffie-Hellman (ℓ-PDDH) assumption) *It holds in $(\mathbb{G}, \mathbb{G}_T)$ if for all PPT adversaries $\mathcal{A}$, the advantage $\boldsymbol{Adv}_{\mathbb{G},\mathbb{G}_T}^{\ell\text{-}PDDH}(k)$:*

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

is negligible in k where $g \xleftarrow{\$} \mathbb{G}^*$, $H, H_1, \ldots, H_\ell \xleftarrow{\$} \mathbb{G}_T^*$ and $\alpha \xleftarrow{\$} \mathbb{Z}_p$.

The ℓ-PDDH assumption is implied by $(\ell+1)$-BDHE assumption [4].

3.2 Building Blocks

In our construction, we employ a fully secure CP-ABE and a fully-simulatable adaptive OT. Below are the definitions of these two building blocks.

CP-ABE. It consists of four PPT algorithms ($Setup_{ABE}$, $GenKey_{ABE}$, Enc_{ABE}, Dec_{ABE}) [2,11]. The setup algorithm, $Setup_{ABE}(1^k)$ generates a master public/secret key pair (pk, mk). The key generation algorithm, $GenKey_{ABE}$ (mk, S), takes the master secret key mk and an attribute set $S \subseteq \mathcal{U}$ outputs a decryption key dk. The encryption algorithm, $Enc_{ABE}(pk, M, \mathbb{A})$, takes pk, a message $M \in \{0,1\}^*$ and an access policy $\mathbb{A}$, produces a ciphertext C. The decryption algorithm, $Dec_{ABE}(pk, C, dk)$, takes pk, decryption key dk and C, outputs M if S associated with dk satisfies $\mathbb{A}$, which is associated with C. The security model (full security) [2,11] is given as follows.

Definition 3. *A CP-ABE is fully secure if for all PPT adversaries $\mathcal{A}$, the advantage of $\mathcal{A}$ in the game below is negligible in k.*

Setup: The challenger runs $Setup_{ABE}(1^k)$ *and gives* pk *to* $\mathcal{A}$.

Phase 1: $\mathcal{A}$ *may query for the decryption keys of attribute sets* $S_1, \ldots, S_{q_1} \subseteq \mathcal{U}$.

Challenge: $\mathcal{A}$ *submits two equal-length messages* $M_0, M_1 \in \{0,1\}^*$ *and a challenge access policy* $\mathbb{A}^*$ *such that none of the sets* $S_1, \ldots, S_{q_1}$ *satisfies* $\mathbb{A}^*$. *The challenger flips a random coin* $b \in \{0,1\}$ *and encrypts* M_b *with respect to* $\mathbb{A}^*$. *The ciphertext* C^* *is given to* $\mathcal{A}$.

Phase 2: Same as Phase 1 with the restriction that none of the additional attribute sets $S_{q_1+1}, \ldots, S_q$ *satisfies* $\mathbb{A}^*$.

Guess: $\mathcal{A}$ *outputs* $b' \in \{0,1\}$. *The advantage of* $\mathcal{A}$ *is defined as* $|\Pr[b' = b] - \frac{1}{2}|$.

Fully-Simulatable Oblivious Transfer. We employ the fully-simulatable adaptive OT due to Camenisch et al. [5]. The sender, on input $k \in \mathbb{N}$ and messages $M_1, \ldots, M_N \in \{0,1\}^*$, randomly generates g, h from $\mathbb{G}^*$ and calculates $H = \hat{e}(g, h)$. It also randomly chooses x from $\mathbb{Z}_p$ and calculates $y = g^x$. The sender's public key $pk = (\hat{e}, \mathbb{G}, \mathbb{G}_T, p, g, y, H)$ and secret key $sk = (h, x)$. For $i = 1, \ldots, N$, the sender calculates $A_i = g^{\frac{1}{x+i}}$, $B_i = \hat{e}(h, A_i) \cdot M_i$ and sets $C_i = (A_i, B_i)$. The sender sends $C_1, \ldots, C_N$ to the receiver along with pk. The sender also shows that $PK\{(h) : H = \hat{e}(g, h)\}$.

When the receiver wants to obliviously transfer a message M_σ where index $\sigma \in [1, N]$, it randomly chooses v from $\mathbb{Z}_p$ and calculates $V = A_\sigma^v$. The receiver sends V along with $PK\{(\sigma, v) : \hat{e}(V, y) = \hat{e}(V, g)^{-\sigma}\hat{e}(g, g)^v\}$ to the sender. The sender verifies it and calculates $W = \hat{e}(h, V)$. It sends W along with $PK\{(h) : H = \hat{e}(g, h) \wedge W = \hat{e}(h, V)\}$ to the receiver. The receiver verifies it and calculates $M_\sigma = \frac{B_\sigma}{W^{1/v}}$.

The security model of Camenisch et al. scheme [5] is fully simulatable, meaning that both the sender and the receiver security are formalized by a simulation-based definition. Full simulatability is required even if the receiver can adaptively choose the message to obliviously transfer, based on those messages it has received from the sender. [5] proves that the above scheme is fully simulatable secure under the $(N+1)$-SDH and the $(N+1)$-PDDH assumptions in the standard model.

Note that $A_i = g^{\frac{1}{x+i}}$ is a modified Boneh-Boyen signature [5] on the message i with the signer's secret key x. [5] mentions that such a signature scheme is unforgeable under the weak chosen message attack [3,5] provided that the $(N+1)$-SDH assumption holds in $\mathbb{G}$, meaning that the receiver in the oblivious transfer protocol cannot forge a valid $A_i = g^{\frac{1}{x+i}}$ by herself.

4 Our Construction

In this section, we first devise a new AC-OT construction (Sec. 4.1) which employs a fully secure CP-ABE. This new AC-OT construction supports the same access policies of the underlying CP-ABE. Then, we discuss the security of the construction. At last, we instantiate the construction (Sec. 4.1) with a concrete CP-ABE.

4.1 A New AC-OT Construction

In our construction, we combine a ciphertext-policy attribute-based encryption with Camenisch et al.'s oblivious transfer to provide the access control and oblivious transfer functionality in an AC-OT protocol. Interestingly, [4] also uses Camenisch et al.'s OT to build their AC-OT construction. However, we should note that the reason why we choose Camenisch et al.'s OT is quite different from [4]. This reason also relates to the idea how we implement access control and why our AC-OT construction can support disjunction policy directly, but [4] cannot.

Recall that given the i-th record, Camenisch et al.'s OT construction generates a modified Boneh-Boyen signature $A_i = g^{\frac{1}{(i+x)}}$ (where x is the secret key). The user proves to the database server in a zero-knowledge fashion that she tries to access the i-th record. [4] extends this idea by embedding a conjunction policy $c_1 \wedge \ldots \wedge c_l$ into $A_i = g^{\frac{1}{(i+x+x_1c_1+\cdots+x_lc_l)}}$ (where now $(x, x_1, \ldots, x_l)$ is the secret key). Then, the user can prove in zero-knowledge fashion that she has the attributes $c_1, \ldots, c_l$ and tries to access the i-th record.

However, it seems hard to find an expression allowing to embed a disjunction policy into A_i. To overcome this problem, we do not embed an access policy into A_i, but instead we introduce ciphertext-policy attribute-based encryption (CP-ABE) to the methodology of devising an AC-OT construction. Specifically, we encrypt $A_i = g^{1/(i+x)}$ using CP-ABE under the access policy of the i-th record and let the database server distribute the ciphertext to the users. Now the user can conditionally release A_i provided that she has a decryption key (which acts as credentials in our construction) associated with attributes satisfy the policy. Then, the user can use the decryption result A_i in a Camenisch's oblivious transfer protocol to obtain record R_i. It is easy to see that such an AC-OT construction supports the same access policy of the underlying CP-ABE, which allows the construction to support disjunction policy directly.

The security of our AC-OT construction relies on the fact that A_i is a modified Boneh-Boyen signature, which cannot be forged by the users. Therefore, the only way to obtain A_i is done by a proper CP-ABE decryption. To our knowledge, the only fully-simulatable adaptive oblivious transfer including a signature in it is the construction proposed by Camenisch et al. [5]. That is the reason why we choose Camenisch et al.'s OT. Since the database setup algorithm is non-interactive, we postpone the proof-of-knowledge $PK\{(h) : H = \hat{e}(g, h)\}$ in [5] to the Transfer protocol (this trick is used in [4] as well.); We also encrypt $C_i = (A_i, B_i)$ in OT rather than distributing them directly to the users. The details of this construction is as follows.

$ISetup(1^k)$: Given a security parameter $k \in \mathbb{N}$, the issuer setup algorithm runs $Setup_{ABE}(1^k)$ to generate a pair of keys (pk_I, sk_I). The issuer publishes pk_I to all parties.

Issue : The user sends f_I, which is initially set to 0, to the issuer. If f_I is 0, the issuer gives $PK\{(sk_I) : (sk_I, pk_I) \text{ is a key pair}\}$ to the user. The user also updates $f_I = 1$. Then, the user sends an attribute set S to the issuer. The issuer runs $dk_S \overset{\$}{\leftarrow} GenKey_{ABE}(sk_I, S)$ to generate a decryption key. The issuer sends dk_S to the user as a credential $Cred_S$ corresponding to S.

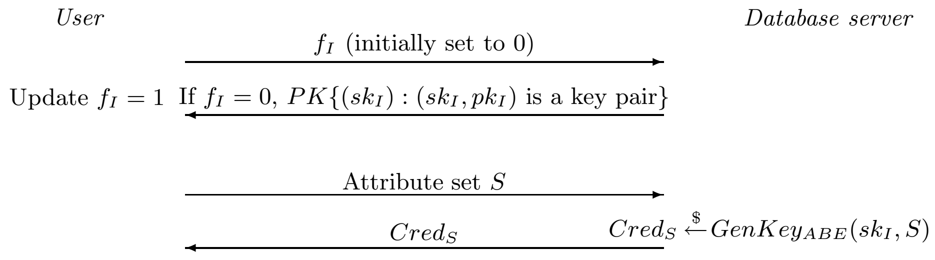
User
Database server
f_I (initially set to 0)
Update $f_I = 1$
If $f_I = 0$, $PK\{(sk_I) : (sk_I, pk_I)$ is a key pair$\}$
Attribute set S
$Cred_S$
$Cred_S \overset{\$}{\leftarrow} GenKey_{ABE}(sk_I, S)$

Fig. 1. Issue Protocol

To prove the issuer's knowledge on her private key is important in our security proof because we need to construct an ideal-world adversary who extracts the private key to decrypt for any CP-ABE ciphertexts.

DBSetup$(pk_I, (R_i, AP_i)_{i=1,\ldots,N})$: The database setup algorithm first chooses $\mathbb{G}$ and $\mathbb{G}_T$ with the same prime order p. It also chooses a bilinear map $\hat{e} : \mathbb{G} \times \mathbb{G} \to \mathbb{G}_T$. It randomly chooses g, h from $\mathbb{G}$ and randomly chooses x from $\mathbb{Z}_p$. It also calculates $H = \hat{e}(g, h)$ and $y = g^x$. For each $i = 1, \ldots, N$, this algorithm calculates $C_i = (A_i, B_i)$ where $A_i = g^{\frac{1}{x+i}}$ and $B_i = \hat{e}(A_i, h)R_i$. It sets the public key $pk_{DB} = (\hat{e}, \mathbb{G}, \mathbb{G}_T, p, g, y, H)$ and the secret key $sk_{DB} = (h, x)$.

Then, it runs $D_i = Enc_{ABE}(pk_I, C_i, AP_i)$ which encrypts C_i under the policy AP_i. The database server publishes $(ER_i)_{i=1,\ldots,N} = (D_i, AP_i)_{i=1,\ldots,N}$ and pk_{DB} to all users. It stores pk_{DB} and keeps sk_{DB} as secret.

Transfer : It is shown by the Fig. 2. The user U first decrypts D_σ to C_σ using $Cred_S$ corresponding to the attribute set S.

Then, U randomly chooses v from $\mathbb{Z}_p$ and calculates $V = A_\sigma^v$. The user U sends V to the database server along with a zero-knowledge proof-of-knowledge $PK\{(v, \sigma) : \hat{e}(V, y) = \hat{e}(V, g)^{-\sigma}\hat{e}(g, g)^v\}$. Then, the database server verifies the proof and calculates $W = \hat{e}(h, V)$ via its $sk_{DB} = (h, x)$. The database server sends W along with $PK\{(h) : H = \hat{e}(g, h) \wedge W = \hat{e}(h, V)\}$ to U. U obtains $R_\sigma = \frac{B_\sigma}{W^{1/v}}$.

4.2 Security

Theorem 1. *The AC-OT protocol described in subsection 4.1 securely implements the AC-OT functionality, provided that the underlying CP-ABE is fully secure, the $(N+1)$-SDH assumption holds in $\mathbb{G}$, the $(N+1)$-PDDH assumption holds in $\mathbb{G}$ and $\mathbb{G}_T$, the knowledge error of the underlying PK is negligible and the underlying PK is perfect zero-knowledgeness.*

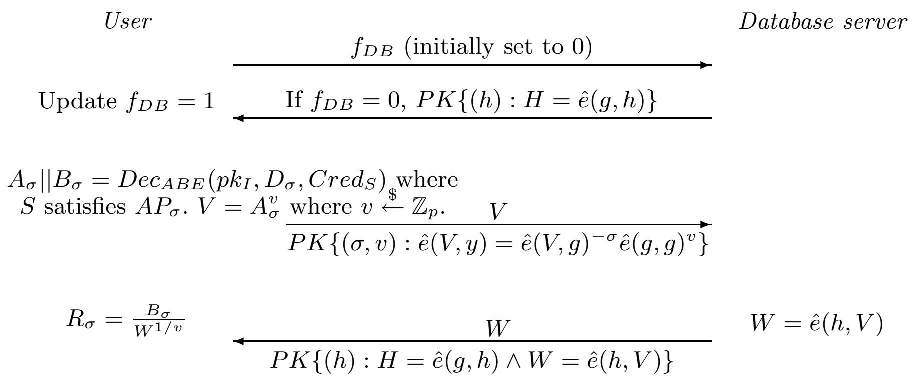
User
Database server
f_{DB} (initially set to 0)
Update $f_{DB} = 1$
If $f_{DB} = 0$, $PK\{(h) : H = \hat{e}(g, h)\}$
$A_\sigma || B_\sigma = Dec_{ABE}(pk_I, D_\sigma, Cred_S)$ where
S satisfies AP_σ. $V = A_\sigma^v$ where $v \xleftarrow{\$} \mathbb{Z}_p$.
V
$PK\{(\sigma, v) : \hat{e}(V, y) = \hat{e}(V, g)^{-\sigma}\hat{e}(g, g)^v\}$
$R_\sigma = \frac{B_\sigma}{W^{1/v}}$
W
$W = \hat{e}(h, V)$
$PK\{(h) : H = \hat{e}(g, h) \wedge W = \hat{e}(h, V)\}$

Fig. 2. Transfer Protocol

We organize the proof to Theorem 1 into different cases. In each case, some of the parties (i.e. the users, the issuer and the database server) are assumed to be corrupted and controlled by an adversary. We do not consider the cases where all parties are honest/dishonest or where the issuer is the only honest/dishonest party, as these cases do not have practical impact.

For each case, we assume that there exists a real-world adversary $\mathcal{A}$ and show how to construct an ideal-world adversary $\mathcal{A}'$ such that no PPT environment Z can tell whether it interacts with $\mathcal{A}$ in the real world or $\mathcal{A}'$ in the ideal world.

Following the strategy of game-hopping, we define a sequence of hybrid games $Game_0$ to $Game_n$. In $Game_i$ $(i = 1, \ldots, n)$, we construct a simulator $\mathcal{S}_i$ that runs $\mathcal{A}$ as a subroutine and provides the view for the environment Z. $Game_0$ models the case of the real world, while in $Game_n$, the simulator $\mathcal{S}_n$ can be used to construct an ideal-world adversary $\mathcal{A}'$. More specifically, the adversary $\mathcal{A}'$ runs $\mathcal{A}$ as a subroutine and provides the same view of $\mathcal{S}_n$ to the adversary $\mathcal{A}$. $\mathcal{A}'$ also simulates the real-world honest parties that communicate with $\mathcal{A}$. We prove that $Game_i$ and $Game_{i+1}$ are indistinguishable for $i = 0$ to $n - 1$, which means $\mathcal{A}$ in the real world ($\mathcal{S}_0$) and $\mathcal{A}'$ in the ideal world ($\mathcal{S}_n$) are indistinguishable by the environment Z. The details of the proof are given in Appendix A.

4.3 Instantiating with Concrete CP-ABE

In the above, we proposed an AC-OT protocol that employs (any fully secure) CP-ABE to encrypt $A_i || B_i$ where $A_i \in \mathbb{G}$ and $B_i \in \mathbb{G}_T$. However, as we aware that message spaces in the most ciphertext-policy attribute-based encryption (CP-ABE) schemes are restricted to $\overline{\mathbb{G}}_T$ (which may or may not be identical to $\mathbb{G}_T$). In this subsection, we first show, given a CP-ABE scheme, how to employ the idea of "hybrid encryption" to devise a new CP-ABE which does not only support the same access policy as the original one but also supports an unbounded message space. More specifically, given a CP-ABE scheme $(Setup_{ABE}, GenKey_{ABE}, Enc_{ABE}, Dec_{ABE})$ and a data encapsulation mechanism (DEM) (Enc_{DEM}, Dec_{DEM}), we construct a new CP-ABE scheme $(Setup, GenKey, Enc, Dec)$ as follows.

1. $Setup(1^k)$: It runs $(pk, mk) \xleftarrow{\$} Setup_{ABE}(1^k)$ and outputs (pk, mk) as the public and master secret key pair.
2. $GenKey(mk, S)$: It runs $dk_S \xleftarrow{\$} GenKey_{ABE}(mk, S)$ and outputs dk_S.
3. $Enc(pk, M, \mathbb{A})$: Given $M \in \{0,1\}^*$, it first randomly chooses K from $\overline{\mathbb{G}}_T$ (i.e., the key space of DEM is assumed to be $\overline{\mathbb{G}}_T$). It computes $C_1 \xleftarrow{\$} Enc_{ABE}(pk, K, \mathbb{A})$. It also calculates $C_2 \xleftarrow{\$} Enc_{DEM}(K, M)$ and outputs the ciphertext $C = (C_1, C_2)$.
4. $Dec(pk, C, dk_S)$: Denote C as (C_1, C_2). It first runs $K{=}Dec_{ABE}(pk, C_1, dk_S)$. Then, it computes $M = Dec_{DEM}(K, C_2)$ and outputs M as the decryption result.

Theorem 2. *The CP-ABE construction described above is fully secure (under CPA), provided that the underlying original CP-ABE scheme is fully secure (under CPA) and DEM is one-time indistinguishability (IND-OT) [9] secure.*

More specifically, given an adversary $\mathcal{A}$ in the full security game of the new CP-ABE, we can construct an adversary $\mathcal{B}_1$ in the full security game of the original CP-ABE and an adversary $\mathcal{B}_2$ in the IND-OT game of DEM and show that $\mathbf{Adv}_{\mathcal{A}}(k) \leq 2\mathbf{Adv}_{\mathcal{B}_1}(k) + \mathbf{Adv}_{\mathcal{B}_2}(k)$. Since the original CP-ABE is fully secure (under CPA) and DEM is IND-OT secure, $\mathbf{Adv}_{\mathcal{B}_1}(k)$ and $\mathbf{Adv}_{\mathcal{B}_2}(k)$ are negligible in security parameter $k \in \mathbb{N}$, which completes the proof. The detailed proof is omitted from the paper.

Since DEM required in Theorem 2 is IND-OT secure, one-time pad is enough. $Enc_{DEM}(K, M)$ can be done by first employing a pseudo-random bit generator G to stretch the K and then outputting the ciphertext as $G(K) \oplus M$ where $\oplus$ is XOR operation. Theoretically, a pseudo-random bit generator can be built from any one-way function. Practically, we can use a block cipher with counter (CRT) mode or output feedback (OFB) mode to do this, provided that we model the block cipher as a pseudo-random permutation.

Next, we employ a concrete CP-ABE construction in [11] to instantiate the AC-OT construction in subsection 4.1. Recall that [11] supports any linear secret sharing scheme (LSSS) access policy $\mathbb{A} = (A, \rho)$ where A is an $l \times n$ matrix and ρ is a map from each row $\boldsymbol{A_x}$ of A to an attribute $\rho(x) \in \mathcal{U}$. Attribute set $S \subseteq \mathcal{U}$ satisfies $\mathbb{A} = (A, \rho)$ if and only if there exist constants $\{\omega_i\}_{\rho(i) \in S}$ such that $\sum_{\rho(i) \in S} \omega_i \boldsymbol{A_i} = (1, 0, \ldots, 0)$. The AC-OT construction instantiated with Lewko et al.'s CP-ABE [11] is as follows.

$ISetup(1^k)$: Given a security parameter $k \in \mathbb{N}$, the issuer setup algorithm chooses a bilinear group $\overline{\mathbb{G}}$ with a composite order $N' = p_1 p_2 p_3$ where p_1, p_2 and p_3 are primes. It chooses $\overline{\mathbb{G}}_T$ with the same order N' and a bilinear map $\overline{e} : \overline{\mathbb{G}} \times \overline{\mathbb{G}} \rightarrow \overline{\mathbb{G}}_T$. It also randomly chooses $\alpha, a \in \mathbb{Z}_{N'}$ and randomly chooses $u \in \overline{\mathbb{G}}_{p_1}$. For each attribute $i \in \mathcal{U}$, it randomly chooses $s_i \in \mathbb{Z}_{N'}$. The public key pk_I is $(N', u, u^a, Y = \overline{e}(u,u)^\alpha, \{T_i = u^{s_i}\}_{i \in \mathcal{U}})$. The (master) secret key sk_I is (α, X_3) where X_3 is a generator of $\overline{\mathbb{G}}_{p_3}$. The issuer publishes pk_I to all parties.

Issue : The user sends f_I, which is initially set to 0, to the issuer. If f_I is 0, the issuer gives $PK\{(\alpha, p_1, p_2, p_3) : Y = \overline{e}(u, u)^\alpha \wedge N' = p_1p_2p_3\}$ to the user.[1] The user also updates $f_I = 1$. Then, the user sends an attribute set S to the issuer. The issuer randomly chooses $t \in \mathbb{Z}_{N'}$ and randomly chooses $\overline{R}_0, R'_0, \overline{R}_i$ from $\overline{\mathbb{G}}_{p_3}$ for $i \in S$. It outputs a credential $Cred_S$ as the decryption key $dk_S = (S, u^\alpha u^{at}\overline{R}_0, u^t R'_0, \{T_i^t \overline{R}_i\}_{i \in S})$. The issuer sends $Cred_S$ to the user.

DBSetup$(pk_I, (R_i, AP_i)_{i=1,\dots,N})$: The database setup algorithm first chooses $\mathbb{G}$ and $\mathbb{G}_T$ with the same prime order p. It also chooses a bilinear map $\hat{e} : \mathbb{G} \times \mathbb{G} \rightarrow \mathbb{G}_T$. It randomly chooses g, h from $\mathbb{G}$ and randomly chooses x' from $\mathbb{Z}_p$. It also calculates $H = \hat{e}(g, h)$ and $y = g^{x'}$. For each $i = 1, \dots, N$, this algorithm calculates $C_i = (A_i, B_i)$ where $A_i = g^{\frac{1}{x'+i}}$ and $B_i = \hat{e}(A_i, h)R_i$. It sets the public key $pk_{DB} = (\hat{e}, \mathbb{G}, \mathbb{G}_T, p, g, y, H)$ and the secret key $sk_{DB} = (h, x')$.

For each $i = 1, \dots, N$, it parses AP_i as (A, ρ) where A is an $l \times n$ matrix and ρ is a map from each row $\boldsymbol{A_x}$ of A to an attribute $\rho(x) \in \mathcal{U}$. Then, it randomly chooses κ from $\overline{\mathbb{G}}_T$. It also randomly chooses a vector $\boldsymbol{\nu} = (s, v_2, \dots, v_n) \in \mathbb{Z}_{N'}^n$. For each row $\boldsymbol{A_x}$ of A, it randomly chooses $r_x \in \mathbb{Z}_{N'}$. It calculates $D_{i,1} = (A, \rho, \kappa\overline{e}(u, u)^{\alpha s}, u^s, \{u^{a\boldsymbol{A_x}\cdot\boldsymbol{\nu}} T_{\rho(x)}^{-r_x}, u^{r_x}\}_x)$. It also runs $Enc_{DEM}(\kappa, A_i||B_i)$ to obtain $D_{i,2}$. Note that $\kappa, \boldsymbol{\nu}$ and r_x (for each x) are chosen freshly for every $i = 1, \dots, N$.

The database server publishes $(ER_i)_{i=1,\dots,N} = (D_i = (D_{i,1}, D_{i,2}), AP_i)_{i=1,\dots,N}$ and pk_{DB} to all users. It stores pk_{DB} and keeps sk_{DB} as secret.

Transfer : Denote $D_{\sigma,1} = (A, \rho, C, C', \{\overline{C}_x, D_x\}_x)$ and $Cred_S = (S, K, L, \{K_i\}_{i \in S})$. The user U first computes constants $\omega_x \in \mathbb{Z}_{N'}$ such that $\sum_{\rho(x) \in S} \omega_x \boldsymbol{A_x} = (1, 0, \dots, 0)$. It also computes $\overline{e}(C', K) / \prod_{\rho(x) \in S} (\overline{e}(\overline{C}_x, L)\overline{e}(D_x, K_{\rho(x)}))^{\omega_x} = \overline{e}(u, u)^{\alpha s}$. Then, it recovers $\kappa = C / \overline{e}(u, u)^{\alpha s}$ and runs $Dec_{DEM}(\kappa, D_{\sigma,2})$ to obtain $C_\sigma = (A_\sigma, B_\sigma)$. Note that if S satisfies (A, ρ), U will find ω_x efficiently [11].

U randomly chooses v from $\mathbb{Z}_p$ and calculates $V = A_\sigma^v$. The user U sends V to the database server along with a zero-knowledge proof-of-knowledge $PK\{(v, \sigma) : \hat{e}(V, y) = \hat{e}(V, g)^{-\sigma} \hat{e}(g, g)^v\}$. Then, the database server verifies the proof and calculates $W = \hat{e}(h, V)$ via its $sk_{DB} = (h, x')$. The database server sends W along with $PK\{(h) : H = \hat{e}(g, h) \wedge W = \hat{e}(h, V)\}$ to U. U obtains $R_\sigma = \frac{B_\sigma}{W^{1/v}}$.

It is easy to see that the security of the above AC-OT construction is a corollary of Theorem 1 and 2.

5 Performance

Given the concrete construction in subsection 4.3, we analyze the encrypted record size for the access policies of CNF formula and threshold gate. We should

[1] Strictly speaking, this is not identical to $PK\{(sk_I) : (sk_I, pk_I)$ is a key pair $\}$. However, to prove $N' = p_1p_2p_3$ would be efficient. More importantly, recall that in the security proof, the constructed ideal-world adversary just needs to extract sk_I from the PK. Given p_1, p_2, p_3, the adversary can generate a valid sk_I by itself.

note that our construction is not restricted to express CNF formula and threshold gate, but any policies which can be expressed by a LSSS matrix.

Given a general CNF formula $(A_{1,1} \vee \ldots \vee A_{1,n_1}) \wedge \ldots \wedge (A_{t,1} \vee \ldots \vee A_{t,n_t})$, we can first represent it by an access tree whose interior nodes are AND and OR gates and leaf nodes are attributes (e.g., $A_{1,1}$). It is easy to see that such an access tree has $n_1 + \ldots + n_t$ leaf nodes.

Lemma 1. *For an access policy which can be expressed by an access tree whose interior nodes are AND and OR gates and leaf nodes are attributes, the ciphertext size in the CP-ABE construction [11] is $O(n)$ where n is the number of leaf nodes in that access tree.*

Lemma 1 is given in [11]. The encrypted (i-th) record D_i consists of two components $D_{i,1}$ and $D_{i,2}$ where $D_{i,1}$ is a CP-ABE ciphertext under the above access tree policy for CNF formula. Therefore, $D_{i,1}$ is of size of $O(n_1 + \ldots + n_t)$. $D_{i,2}$ is a DEM ciphertext whose size is a constant (e.g, $O(1)$). To sum up, the encrypted record size in our AC-OT construction is $O(\sum_{i=1}^{t} n_i)$. We also show a comparison for our construction and [4] (and [6]) in Table 1 for completeness. Note that due to the use of duplication strategy in [4], the record of above CNF formula will appear for $n_1 \times \ldots \times n_t$ times.

Table 1. Comparison of AC-OT protocols expressing CNF formula

Protocol	Encrypted record size
[6]	$O(2^{\sum_{i=1}^{t} n_i})$
[4]	$O(\prod_{i=1}^{t} n_i)$
This paper	$O(\sum_{i=1}^{t} n_i)$

Given a threshold gate T_k^n: with at least k attributes in the set $\{A_1, \ldots, A_n\}$, we have the following lemma.

Lemma 2. *For any $1 \leq k \leq n$, a LSSS matrix A for threshold gate T_k^n can be constructed with n rows.*

Specifically, we can construct a $n \times k$ matrix $A = (\boldsymbol{a_1}, \ldots, \boldsymbol{a_n})^T$ where $\boldsymbol{a_i}$ ($i = 1, \ldots, n$) are k-length vectors such that any k of them consist of a base. Therefore, if more than k attributes (without loss of generality, we assume the attributes correspond to $\boldsymbol{a_{i_1}}, \ldots, \boldsymbol{a_{i_k}}, \ldots, \boldsymbol{a_{i_j}}$), then we can find $\omega_1, \ldots, \omega_k$ such that $(1, 0, \ldots, 0) = \sum_{l=1}^{k} \omega_k \boldsymbol{a_{i_l}}$. However, if there are less than k attributes, $\boldsymbol{a_{i_1}}, \ldots, \boldsymbol{a_{i_j}}$ ($j < k$) is linear independent with $(1, 0, \ldots, 0)$. The proof to Lemma 2 is omitted in this paper.

In the construction (Sec. 4.3), the size of $D_{i,1}$ is $O(n)$ if the LSSS matrix A has n rows. Consequently, the encrypted record size of a T_k^n threshold gate is $O(n)$ in our AC-OT construction. The comparison for our construction and [4] (and [6]) is shown in Table 2. The record will appear for $\binom{n}{k}$ times in [4] due to its duplication strategy.

Table 2. Comparison of AC-OT protocols expressing threshold gate

Protocol	Encrypted record size
[6]	$O(2^n)$
[4]	$O(\binom{n}{k})$
This paper	$O(n)$

6 Conclusion

In this paper, we proposed a new AC-OT construction which is secure in the standard model. Our construction is based on an observation that Camenisch et al.'s OT construction contains an unforgeable signature, which allows us to conditionally release the signature with a ciphertext-policy attribute-based encryption. Without duplicating records, our construction reduces the size of the encrypted database by a substantial amount for access policies represented in CNF or k-of-n threshold gate.

Acknowledgments

This work was supported by Grant HKU 715509E from Hong Kong RGC. We thank the anonymous reviewers for their valuable comments.

References

1. Beimel, A.: Secure schemes for secret sharing and key distribution. PhD thesis, Israel Institute of Technology, Technion, Haifa, Israel (1996)
2. Bethencourt, J., Sahai, A., Waters, B.: Ciphertext-policy attribute-based encryption. In: 28th IEEE Symposium on Security and Privacy, pp. 321–334. IEEE Press, New York (2007)
3. Boneh, D., Boyen, X.: Short signatures without random oracles. In: Cachin, C., Camenisch, J. (eds.) EUROCRYPT 2004. LNCS, vol. 3027, pp. 56–73. Springer, Heidelberg (2004)
4. Camenisch, J., Dubovitskaya, M., Neven, G.: Oblivious transfer with access control. In: 16th ACM Conference on Computer and Communications Security, pp. 131–140. ACM, New York (2009)
5. Camenisch, J., Neven, G., Shelat, A.: Simulatable adaptive oblivious transfer. In: Naor, M. (ed.) EUROCRYPT 2007. LNCS, vol. 4515, pp. 573–590. Springer, Heidelberg (2007)
6. Coull, S., Green, M., Hohenberger, S.: Controlling access to an oblivious database using stateful anonymous credentials. In: Jarecki, S., Tsudik, G. (eds.) PKC 2009. LNCS, vol. 5443, pp. 501–520. Springer, Heidelberg (2009)
7. Google Inc.: Google health, `https://www.google.com/health`
8. Green, M., Hohenberger, S.: Practical adaptive oblivious transfer from a simple assumption. Cryptology ePrint Archive, Report 2010/109 (2010), `http://eprint.iacr.org/`

9. Herranz, J., Hofheinz, D., Kiltz, E.: KEM/DEM: Necessary and sufficient conditions for secure hybrid encryption. Cryptology ePrint Archive, Report 2006/265 (2006), `http://eprint.iacr.org/`
10. Jarecki, S., Liu, X.: Efficient oblivious pseudorandom function with applications to adaptive OT and secure computation of set intersection. In: Reingold, O. (ed.) TCC 2009. LNCS, vol. 5444, pp. 577–594. Springer, Heidelberg (2009)
11. Lewko, A., Okamoto, T., Sahai, A., Takashima, K., Waters, B.: Fully secure functional encryption: Attribute-based encryption and (hierarchical) inner product encryption. In: Gilbert, H. (ed.) EUROCRYPT 2010. LNCS, vol. 6110, pp. 62–91. Springer, Heidelberg (2010), full version `http://eprint.iacr.org/2010/110`

A Proof of Theorem 1

A.1 Proof of Case 1 (The Database Server and the Issuer Are Dishonest)

Lemma 3. *For all environments Z and any real-world adversary $\mathcal{A}$ controlling the database server and the issuer, there exists an ideal-world adversary $\mathcal{A}'$ such that the probability that Z can distinguish whether it interacts with $\mathcal{A}$ in the real world or it interacts with $\mathcal{A}'$ in the ideal world is negligible, provided that the knowledge error of the underlying PK is negligible and the underlying PK is perfect zero-knowledgeness.*

$Game_0$. The real world. $\mathcal{S}_0$ plays the role of the honest users.

$Game_1$. $\mathcal{S}_1$ is the same as $\mathcal{S}_0$ except that at the first time of issue query instructed by environment Z, $\mathcal{S}_1$ runs the extractor of $PK\{(sk_I) : sk_I$ and pk_I is a key pair$\}$ to extract sk_I from $\mathcal{A}$. If it fails, $\mathcal{S}_1$ outputs $\perp$ to Z. Recall that sk_I and pk_I is the (master) secret key and the public key of the underlying CP-ABE.

Note that the difference between $Game_0$ and $Game_1$ is negligible provided that the underlying PK is sound. We believe that constructing such PK for most CP-ABE should be easy.

$Game_2$. $\mathcal{S}_2$ is the same as $\mathcal{S}_1$ except that at the first time of transfer query instructed by environment Z, $\mathcal{S}_2$ runs the extractor of $PK\{(h) : H = \hat{e}(g,h)\}$ to extract h from $\mathcal{A}$. If it fails, $\mathcal{S}_2$ outputs $\perp$ to Z.

The difference between $Game_1$ and $Game_2$ is negligible provided that the underlying PK is sound.

$Game_3$. $\mathcal{S}_3$ is the same as $\mathcal{S}_2$ except that at each time of transfer query instructed by environment Z, $\mathcal{S}_3$ engages in Transfer protocol with $\mathcal{A}$ to query a record randomly chosen from those which it has the necessary decryption keys, rather than querying σ instructed by Z.

Note that $Game_2$ and $Game_3$ is identical due to the perfect zero-knowledgeness of the underlying $PK\{(v,\sigma) : \hat{e}(V,y) = \hat{e}(V,g)^{-\sigma}\hat{e}(g,g)^v\}$.

Now we show how to construct the ideal-world adversary $\mathcal{A}'$ with black-box access to $\mathcal{A}$, where $\mathcal{A}'$ incorporates all steps from $Game_3$.

Adversary $\mathcal{A}'$ first runs $\mathcal{A}$ to obtain $\{ER_i\}$ and public parameters. At the each issue query instructed by Z, $\mathcal{A}'$ will simulate a user interacting with $\mathcal{A}$ for issuing the decryption key. If the decryption key is valid, $\mathcal{A}'$ will send $b = 1$ to T; otherwise, it will send $b = 0$. If it is the first time of Issue protocol, $\mathcal{A}'$ will also run the extractor with $\mathcal{A}$ to extract sk_I of the underlying CP-ABE (As we have shown, the probability that the extractor fails is negligible).

When receiving $(transfer)$ from T, $\mathcal{A}'$ will query a record randomly chosen from those which it has the necessary decryption keys with $\mathcal{A}$. If the Transfer protocol succeeds, $\mathcal{A}'$ will send back $b = 1$ to T; otherwise, it sends back $b = 0$. If it is the first time of transfer query instructed by Z, $\mathcal{A}'$ will also run the extractor to extract h with $\mathcal{A}$ (As we have shown, the extractor fails with only negligible probability).

If $\mathcal{A}'$ has extracted sk_I and h from $\mathcal{A}$, $\mathcal{A}'$ will use sk_I to decrypt ER_i to $A_i||B_i$ where $B_i = \hat{e}(h, A_i)R_i$. Then, $\mathcal{A}'$ calculates $R_i = \frac{B_i}{\hat{e}(h,A_i)}$. $\mathcal{A}'$ sends $(R_i, AP_i)_{i=1,\ldots,N}$ to T for initDB.

A.2 Proof of Case 2 (Only the Database Server Is Dishonest)

Lemma 4. *For all environments Z and any real-world adversary $\mathcal{A}$ controlling the database server, there exists an ideal-world adversary $\mathcal{A}'$ such that the probability that Z can distinguish whether it interacts with $\mathcal{A}$ in the real world or it interacts with $\mathcal{A}'$ in the ideal world is negligible, provided that the knowledge error of the underlying PK is negligible and the underlying PK is perfect zero-knowledgeness.*

Game$_0$. The real world. $\mathcal{S}_0$ plays the role of the honest users and the issuer to $\mathcal{A}$.

Game$_1$. $\mathcal{S}_1$ is the same as $\mathcal{S}_0$ except that at the first time of transfer query, $\mathcal{S}_1$ runs the extractor of $PK\{(h) : H = \hat{e}(g, h)\}$ with $\mathcal{A}$ to extract h. If it is failed, $\mathcal{S}_1$ outputs $\perp$ to Z.

The difference between *Game*$_0$ and *Game*$_1$ is negligible provided that the underlying PK is sound.

Game$_2$. $\mathcal{S}_2$ is the same as $\mathcal{S}_1$ except that at each time of transfer query, $\mathcal{S}_2$ engages in the Transfer protocol with $\mathcal{A}$ to query a record randomly chosen from those which it has the necessary decryption keys, rather than querying σ instructed by Z.

Game$_1$ is identical *Game*$_2$ due to the zero-knowledgeness of the underlying $PK\{(v, \sigma) : \hat{e}(V, y) = \hat{e}(V, g)^{-\sigma}\hat{e}(g, g)^v\}$.

Now we show how to construct the ideal-world adversary $\mathcal{A}'$. Adversary $\mathcal{A}'$ first runs $\mathcal{A}$ to obtain $\{ER_i\}$ and public parameters. At each time of transfer query, $\mathcal{A}'$ simulates a user interacting with $\mathcal{A}$ in Transfer protocol to query a record randomly chosen from those which it has the necessary decryption keys. If Transfer protocol succeeds, $\mathcal{A}'$ sends $b = 1$ to T; otherwise, it sends $b = 0$. If it is the first time of transfer query, $\mathcal{A}'$ also runs the extractor of

$PK\{(h) : H = \hat{e}(g,h)\}$ with $\mathcal{A}$ to extract h (As we have shown that the extractor fails with only negligible probability).

If $\mathcal{A}'$ has extracted h from $\mathcal{A}$, $\mathcal{A}'$ (since it also simultaneously simulates the issuer) uses its (master) secret key sk_I to decrypt $\{ER_i\}$ to $A_i||B_i$ where $B_i = \hat{e}(h, A_i)R_i$. $\mathcal{A}'$ calculates $R_i = \frac{B_i}{\hat{e}(h,A_i)}$. $\mathcal{A}'$ also sends $(R_i, AP_i)_{i=1,\ldots,N}$ to T for initDB.

A.3 Proof of Case 3 (Only Some Users Are Dishonest)

Lemma 5. *For all environments Z and any real-world adversary $\mathcal{A}$ controlling some users, there exists an ideal-world adversary $\mathcal{A}'$ such that the probability that Z can distinguish whether it interacts with $\mathcal{A}$ in the real world or it interacts with $\mathcal{A}'$ in the ideal world is negligible, provided that the underlying CP-ABE is fully secure, the $(N+1)$-SDH assumption and the $(N+1)$-PDDH assumption hold, the knowledge error of the underlying PK is negligible and the underlying PK is perfect zero-knowledgeness.*

$Game_0$. The real world. $\mathcal{S}_0$ plays the role of the honest issuer and database server to $\mathcal{A}$.

$Game_1$. $\mathcal{S}_1$ follows the specification except during each transfer query, when $\mathcal{A}$ requests for a record R_σ from the database server on behalf of some user. $\mathcal{A}$ submits V along with the proof of v, σ:

$$PK\{(v,\sigma) : \hat{e}(V,y) = \hat{e}(V,g)^{-\sigma}\hat{e}(g,g)^v\}.$$

$\mathcal{S}_1$ extracts v and σ and continues the rest of the simulation only if extraction is successful. Otherwise, it outputs $\perp$ to Z.

The difference between $Game_0$ and $Game_1$ is negligible provided that the underlying PK is sound.

$Game_2$. $\mathcal{S}_2$ is the same as $\mathcal{S}_1$ except after v, σ has been extracted, $\mathcal{S}_2$ computes $\hat{A}_\sigma = V^{1/v}$. If $\mathcal{A}$ has never request the decryption key for attribute set S satisfying AP_σ from the issuer and that $\hat{A}_\sigma = A_\sigma$ (notice that $\mathcal{S}_2$ simulates the real-world honest database server to adversary $\mathcal{A}$ and $\mathcal{S}_2$ generates $A_1, \ldots, A_N$ by herself.), $\mathcal{S}_2$ outputs $\perp$ to Z.

The difference between $Game_1$ and $Game_2$ is negligible provided that the underlying CP-ABE is fully secure and the $(N+1)$-SDH assumption holds.

If $\mathcal{S}_2$ obtains the correct A_σ from $\mathcal{A}$ such that $\mathcal{A}$ is not given the corresponding CP-ABE decryption key, we can use $\mathcal{A}$ to break the full security of the underlying CP-ABE. More specifically, we show how to construct an adversary $\mathcal{B}$ that wins the full security game of the underlying CP-ABE. $\mathcal{B}$ plays the role of $\mathcal{S}_2$ and is with black-box access to $\mathcal{A}$.

The challenger first runs $Setup_{ABE}(1^k)$ and gives the public key pk to $\mathcal{B}$. $\mathcal{B}$ submits AP_σ to the challenger where AP_i ($i = 1, \ldots, N$) is instructed by the environment Z. $\mathcal{B}$ also computes $A_i = g^{\frac{1}{x+i}}$ and $B_i = \hat{e}(h, A_i)R_i$ for $i = 1, \ldots, N$

(Note that $\mathcal{B}$ runs Z as subroutine and simulates the real-world honest database server.). Then $\mathcal{B}$ randomly chooses two random numbers r_A from $\mathbb{G}$ and r_B from $\mathbb{G}_T$. $\mathcal{B}$ sets $M_0 = A_\sigma || B_\sigma$ and $M_1 = r_A || r_B$. $\mathcal{B}$ submits M_0 and M_1 to the challenger. The challenger flips a coin $b \in \{0, 1\}$ and encrypts M_b to C^* under AP_σ. The ciphertext C^* is given to $\mathcal{B}$. $\mathcal{B}$ also encrypts $A_i || B_i$ to ciphertext C_i under AP_i for all $i = 1, \ldots, \sigma - 1, \sigma + 1, \ldots, N$. $\mathcal{B}$ publishes $(C_1, AP_1), \ldots, (C_{\sigma-1}, AP_{\sigma-1}), (C^*, AP_\sigma), (C_{\sigma+1}, AP_{\sigma+1}), \ldots, (C_N, AP_N)$ to $\mathcal{A}$. When $\mathcal{A}$ asks for decryption keys, $\mathcal{B}$ will forward the requests to the challenger. Recall that $\mathcal{A}$ does not request the decryption key for attribute set S satisfying AP_σ from the issuer, the challenger will answer all requests properly. Finally, if $\mathcal{A}$ outputs A_σ, then $\mathcal{B}$ will output $b' = 0$ as its guess bit; otherwise, $\mathcal{B}$ will output $b' = 1$.

When $b = 0$ ($\Pr[b = 0] = \frac{1}{2}$), C^* is the encryption of $A_\sigma || B_\sigma$ and this is identical to $Game_2$. In this case, $\mathcal{A}$ will output A_σ with a non-negligible probability ϵ. When $b = 1$ ($\Pr[b = 1] = \frac{1}{2}$), C^* is encryption of random $r_A || r_B$ which is independent with A_σ. $\mathcal{A}$ would not output proper A_σ, otherwise, $\mathcal{A}$ will forge a modified Boneh-Boyen signature under weak chosen message attack. This happens with a negligible probability η under the $(N + 1)$-SDH assumption [5].

$$\begin{aligned}\Pr[b = b'] &= \Pr[b = 0]\Pr[b' = 0] + \Pr[b = 1]\Pr[b' = 1] \\ &= \frac{1}{2}\epsilon + \frac{1}{2}(1 - \eta) \\ &= \frac{1}{2} + \frac{1}{2}(\epsilon - \eta)\end{aligned}$$

where $\frac{1}{2}(\epsilon - \eta)$ is non-negligible.

Game$_3$. $\mathcal{S}_3$ is the same as $\mathcal{S}_2$ except for each transfer query, it computes $W = (B_\sigma / R_\sigma)^v$ and that the proof-of-knowledge protocol

$$PK\{(h) : W = \hat{e}(V, h)\}$$

becomes a simulated proof such that $\mathcal{S}_3$ does not require the knowledge of h.

The difference between $Game_2$ and $Game_3$ is negligible provided that the underlying PK is perfect zero-knowledge.

Game$_4$. $\mathcal{S}_4$ is the same as $\mathcal{S}_3$ except $\mathcal{S}_4$ now deviates from the database setup protocol by replacing the value B_i from $\hat{e}(h, A_i)R_i$ to just random elements in group $\mathbb{G}_T$.

The difference between $Game_3$ and $Game_4$ is negligible provided that the $(N + 1)$-PDDH assumption holds.

Suppose there exists an environment Z that can distinguish $Game_3$ and $Game_4$, we show how to construct an adversary $\mathcal{B}$ that solves the $(N+1)$-PDDH problem. $\mathcal{B}$ is given the problem instance $g', g'^x, g'^{x^2}, \ldots, g'^{x^{N+1}}, H, H_1, \ldots, H_{N+1}$ and its task is to tell if $H_i = H^{x^i}$ or H_i are just random elements from $\mathbb{G}_T$ for $i = 1, \ldots, N + 1$.

$\mathcal{B}$ runs the environment Z and the adversary $\mathcal{A}$ as subroutines. It plays the role of the honest issuer and database server with $\mathcal{A}$ and Z. Denote $f(x) = \prod_{i=1}^{N}(x+i)$ as the N-degree polynomial in x. Under the additive notation, $f(x) = \sum_{i=0}^{N} \beta_i x^i$. Set $g = g'^{f(x)}$ and thus $g = \prod_{i=0}^{N} (g'^{x^i})^{\beta_i}$ and is computable by $\mathcal{B}$ without the knowledge of x. Next, the value $y = g^x$ is also computable as $g'^{xf(x)}$ since $xf(x)$ is a degree $N+1$ polynomial. $\mathcal{B}$ sets the public key of the database server as $pk_{DB} = \{g, H, y, \hat{e}, \mathbb{G}, \mathbb{G}_T, p\}$

When it is instructed by environment Z to create a database with records $(R_i, AP_i)_{i=1,\ldots,N}$, it generates $A_i = g^{\frac{1}{x+i}}$ by $A_i = g'^{\frac{f(x)}{x+i}}$. This is computable by $\mathcal{B}$ without the knowledge of x since $f(x)/(x+i) = (x+1)\cdots(x+i-1)(x+i+1)\cdots(x+N)$ is a polynomial of degree $N-1$. Next, it computes B_i as $(\prod H_i^{\beta_i})R_i$.

Note that if $H_i = H^{x^i}$ for $i = 1, \ldots, N$, then $\mathcal{B}$ is acting as $\mathcal{S}_3$ in $Game_3$ while if H_i are just random elements in $\mathbb{G}_T$, $\mathcal{B}$ is acting as $\mathcal{S}_4$ in $Game_4$. Thus, if the environment Z can distinguish between $Game_3$ and $Game_4$, we have solved the $(N+1)$-PDDH problem.

Now we show how to construct the ideal-world adversary $\mathcal{A}'$ which is given black-box access to $\mathcal{A}$.

In the beginning, Z tells the ideal-world honest database server to initialize the database with inputs (R_i, AP_i) for $i = 1$ to N. The honest database server forwards this request to the trusted party T who sets database as $(R_i, AP_i)_{i=1,\ldots,N}$. While AP_i is made public to all ideal world users, R_i is kept secret. T also maintains a set S_U to record what set of attributes has been issued to the user U.

$\mathcal{A}'$ setups the key pairs for the issuer as well as the database server to provide the simulation for $\mathcal{A}$. However, it has to setup the database without knowing R_i. $\mathcal{A}'$ did so by setting ER_i as (C_i, AP_i), where C_i is the encryption of $A_i||B_i$, with B_i being random element in $\mathbb{G}_T$. (As we have shown, Z cannot distinguish this difference.)

When Z issues the message $(issue, S)$ to $\mathcal{A}'$, $\mathcal{A}'$ sends the same message to $\mathcal{A}$ (Recall that $\mathcal{A}'$ acts as the environment to $\mathcal{A}$). If $\mathcal{A}$ deviates from the protocol and does nothing (remember $\mathcal{A}'$ also acts as the issuer to $\mathcal{A}$), $\mathcal{A}'$ also did not send any message to T. Otherwise, if $\mathcal{A}$ requests a certain set of credentials from $\mathcal{A}'$, $\mathcal{A}'$ requests the same set of credentials from the trusted party T. If T sends back 1 to $\mathcal{A}'$, $\mathcal{A}'$ issues the corresponding credential to $\mathcal{A}$. On the other hand, if T returns 0, $\mathcal{A}'$ returns 0 to $\mathcal{A}$ as well.

When Z issues the message $(transfer, \sigma)$ to $\mathcal{A}'$, $\mathcal{A}'$ sends the same message to $\mathcal{A}$.If $\mathcal{A}$ deviates from the protocol and does nothing ($\mathcal{A}'$ also acts as the issuer to $\mathcal{A}$), $\mathcal{A}'$ also did not send any message to T. Otherwise, $\mathcal{A}'$ extracts σ from $\mathcal{A}$. (As we have shown, $\mathcal{A}'$ fails with a negligible probability.)

$\mathcal{A}'$ requests for credentials from T that satisfy AP_σ (if there is no any user satisfying this policy). After that, $\mathcal{A}'$ requests for the record R_σ on behalf of that user. Next, it computes $W = (B_\sigma/R_\sigma)^v$ and sends W, along with the zero-knowledge proof, back to $\mathcal{A}$ on behalf of the database server. (Again, it is different from the actual honest database server. However, we have shown that $\mathcal{A}$ (and therefore Z) cannot notice this difference.)

In fact, one can notice that the simulation provided by $\mathcal{A}'$ to $\mathcal{A}$ is the same as the simulator $\mathcal{S}_5$ provided to $\mathcal{A}$. According to the above proof, Z cannot tell any difference between $\mathcal{A}$ and $\mathcal{A}'$.

A.4 Proof of Case 4 (The Issuer and Some Users Are Dishonest)

Lemma 6. *For all environments Z and any real-world adversary $\mathcal{A}$ controlling the issuer and some users, there exists an ideal-world adversary $\mathcal{A}'$ such that the probability that Z can distinguish whether it interacts with $\mathcal{A}$ in the real world or it interacts with $\mathcal{A}'$ in the ideal world is negligible, provided that the $(N+1)$-PDDH assumption holds, the knowledge error of the underlying PK is negligible and the underlying PK is perfect zero-knowledgeness.*

$Game_0$. The real world. $\mathcal{S}_0$ plays the role of honest database server and some honest users to $\mathcal{A}$.

$Game_1$. $\mathcal{S}_1$ is the same as $\mathcal{S}_0$ except that at each time of transfer query, when $\mathcal{A}$ requests for a record R_σ from the database server on behalf of some user. $\mathcal{A}$ submits V along with the proof of v, σ:

$$PK\{(v,\sigma) : \hat{e}(V,y) = \hat{e}(V,g)^{-\sigma}\hat{e}(g,g)^{v}\}.$$

$\mathcal{S}_1$ extracts v and σ. If $\mathcal{S}_1$ fails, it outputs $\perp$ to the environment Z.

The difference between $Game_0$ and $Game_1$ is negligible provided that the underlying PK is sound.

$Game_2$. $\mathcal{S}_2$ is the same as $\mathcal{S}_1$ except that for each time of transfer query, it computes $W = (B_\sigma/R_\sigma)^v$ and that the proof-of-knowledge protocol

$$PK\{(h) : W = \hat{e}(V,h)\}$$

becomes a simulated proof such that $\mathcal{S}_2$ does not require any knowledge of h.

The difference between $Game_1$ and $Game_2$ is negligible provided that the underlying PK is perfect zero-knowledge.

$Game_3$. $\mathcal{S}_3$ is the same as $\mathcal{S}_2$ except that $\mathcal{S}_3$ now deviates from the database setup protocol by replacing the value B_i from $\hat{e}(h, A_i)R_i$ to just random element in $\mathbb{G}_T$.

As we have shown, the difference between $Game_2$ and $Game_3$ is negligible provided that the $(N+1)$-PDDH assumption holds.

Now we show how to construct the ideal-world adversary $\mathcal{A}'$ with black-box access to $\mathcal{A}$. At each time of transfer query, $\mathcal{A}'$ extracts (v,σ) from $\mathcal{A}$ (As we have shown it fails with only negligible probability). $\mathcal{A}'$ requests the record R_σ on behalf of the user. $\mathcal{A}'$ queries T for the decryption key of attributes satisfying AP_σ. Since $\mathcal{A}'$ simultaneously acts as the dishonest issuer, $\mathcal{A}'$ will send back $b = 1$ to T for granting the decryption key. Next, $\mathcal{A}'$ computes $W = (B_\sigma/R_\sigma)^v$ and sends W, along with the simulated zero-knowledge proof, back to $\mathcal{A}$ on behalf of the database server (It is different from the actual honest database server. However, as we have shown that $\mathcal{A}$ (and therefore Z) cannot notice the difference.).

$\mathcal{A}'$ setups the key pair of the database server to provide the simulation for $\mathcal{A}$. However, it has to setup the database without knowing R_i. $\mathcal{A}'$ did so by setting $ER_i = (C_i, AP_i)$, where C_i is the encryption of $A_i||B_i$, with B_i being random element in $\mathbb{G}_T$. As we have shown that Z cannot distinguish this difference.

B Lewko et al.'s Fully-Secure CP-ABE [11]

Setup$(\lambda, \mathcal{U})$*:* The setup algorithm chooses a bilinear group $\mathbb{G}$ of order $N = p_1p_2p_3$ where p_1, p_2 and p_3 are 3 distinct primes. Let $\mathbb{G}_{p_i}$ denote the subgroup of order p_i in $\mathbb{G}$. It then chooses random exponents $\alpha, a \in \mathbb{Z}_N$ and a random group element $g \in \mathbb{G}_{p_1}$. For each attribute $i \in \mathcal{U}$, it chooses a random value $s_i \in \mathbb{Z}_N$. The public parameters $PK = (N, g, g^a, \hat{e}(g,g)^\alpha, \{T_i = g^{s_i}\}_{\forall i})$. The master secret key $MSK = (\alpha, X_3)$ where X_3 is a generator of $\mathbb{G}_{p_3}$.

KeyGen(MSK, S, PK)*:* The key generation algorithm chooses a random $t \in \mathbb{Z}_N$ and random elements $R_0, R'_0, R_i \in \mathbb{G}_{p_3}$. The secret key is:

$$SK = (S, K = g^\alpha g^{at} R_0, L = g^t R'_0, \{K_i = T_i^t R_i\}_{i \in S}).$$

Encrypt$((A, \rho), PK, M)$*:* A is an $\ell \times n$ matrix and ρ is map from each row $\boldsymbol{A_x}$ of A to an attribute $\rho(x)$. The encryption algorithm chooses a random vector $v \in \mathbb{Z}_N^n$, denoted $\boldsymbol{v} = (s, v_2, \ldots, v_n)$. For each row A_x of A, it chooses a random $r_x \in \mathbb{Z}_N$. The ciphertext is:

$$CT = (C = M\hat{e}(g,g)^{\alpha s}, C' = g^s, \{C_x = g^{a\boldsymbol{A_x}\cdot\boldsymbol{v}}, D_x = g^{r_x}\}_x).$$

Decrypt(CT, PK, SK)*:* The decryption algorithm computes constants $\omega_x \in \mathbb{Z}_N$ such that $\sum_{\rho(x)\in S} \omega_x \boldsymbol{A_x} = (1, 0, \ldots, 0)$. It then computes:

$$\hat{e}(C', K) / \prod_{\rho(x)\in S} (\hat{e}(C_x, L)\hat{e}(D_x, K_{\rho(x)}))^{\omega_x} = \hat{e}(g,g)^{\alpha s}.$$

Then M can be recovered as $C/\hat{e}(g,g)^{\alpha s}$.