Adaptive Oblivious Transfer with Hidden Access Policy Realizing Disjunction

Vandana Guleria and Ratna Dutta
Department of Mathematics, Indian Institute of Technology Kharagpur, Kharagpur-721302, India

Keywords: Oblivious Transfer, Access Policy, Attribute based Encryption, Full Simulation Security Model.

Abstract: We propose an efficient adaptive oblivious transfer protocol with hidden access policies. This scheme allows a receiver to anonymously recover a message from a database which is protected by hidden attribute based access policy if the receiver's attribute set satisfies the associated access policy implicitly. The proposed scheme is secure in the presence of malicious adversary under the q-Strong Diffie-Hellman (SDH), q-Power Decisional Diffie-Hellman (PDDH) and Decision Bilinear Diffie-Hellman (DBDH) assumption in full-simulation security model. The scheme covers disjunction of attributes. The proposed protocol outperforms the existing similar schemes in terms of both communication and computation.

1 INTRODUCTION

The *adaptive oblivious transfer with hidden access policy* (AOT-HAP) is a widely used primitive in cryptography. It is useful whenever a large database is queried adaptively. The database may contain some sensitive information which the database holder wants to make available only to selected recipients. In AOT-HAP, each message is associated with some access policy (AP). The AP could be attributes, roles, or rights. It represents which combination of attributes a receiver should have in order to access the message. For instance, consider the medical database of patients. Many patients do not want to reveal information about their disease to anybody except their concerned doctors. Also, sometimes doctors may want to remain anonymous. The AOT-HAP is a solution to such type of privacy problems.

The AOT-HAP consists of a sender, an issuer and a set of receivers. The sender has a database of messages. Each message is associated with an access policy. The sender encrypts each message and makes the encrypted database public, while keeping the access policy AP of each message hidden. A receiver with a set of attributes w interacts with the issuer to obtain the attribute secret key for w and decrypts the corresponding message. The receiver can recover the message correctly if its attribute set w implicitly satisfies the access policy AP. The AOT-HAP thus completes in two phases– *initialization phase* and *transfer phase*. In initialization phase, the sender encrypts N messages. In transfer phase, a receiver interacts with sender and issuer adaptively and recovers k messages of its choice, one message in one of the k transfer phases. The sender does not learn which k messages are learnt by which receiver and a receiver remains oblivious about the $N-k$ messages which it did not query. Moreover, the access policies are kept hidden in the encrypted database and a receiver learns nothing about the access policy of a decrypted message during a successful decryption.

Related Work. The oblivious transfer protocol (Camenisch et al., 2007), (Green and Hohenberger, 2007) and (Naor and Pinkas, 1999) allows receivers to access the content of the database without putting any restriction on who can access which message. To introduce this property, Coull *et al.* (Coull et al., 2009) and Camenisch *et al.* (Camenisch et al., 2009) proposed oblivious transfer with access policy. The sender assigned an access policy to each message. Receivers whose attribute sets satisfy the access policy associated with a message can only recover the message. The access policy in (Camenisch et al., 2009) is restricted to conjunction of attributes only (e.g $a_1 \wedge a_2$, where a_1 and a_2 are attributes). To address disjunctive policy, the same message is duplicated, i.e, if m is the message associated with access policy $(a_1 \wedge a_2) \vee (a_3 \wedge a_4)$ then m is be encrypted twice– once with access policy $(a_1 \wedge a_2)$ and once with access policy $(a_3 \wedge a_4)$, where a_1, a_2, a_3 and a_4 are attributes. To overcome this, Zhang *et al.* (Zhang et al., 2010) proposed oblivious transfer

Guleria V. and Dutta R..
Adaptive Oblivious Transfer with Hidden Access Policy Realizing Disjunction.
DOI: 10.5220/0005016900430054
In *Proceedings of the 11th International Conference on Security and Cryptography* (SECRYPT-2014), pages 43-54
ISBN: 978-989-758-045-1

with access control realizing disjunction without duplication. However, the access policies are not hidden in (Camenisch et al., 2009), (Coull et al., 2009) and (Zhang et al., 2010). Recently, (Camenisch et al., 2012) and (Camenisch et al., 2011) proposed AOT-HAP which are to the best of our knowledge the only oblivious transfer protocols with hidden access policies.

Our Contribution. Motivated by the work of (Camenisch et al., 2012) and (Camenisch et al., 2011) which cover only conjunction of attributes, we construct an efficient adaptive oblivious transfer with hidden access policy (AOT-HAP). To the best of our knowledge, our scheme is the *first* AOT-HAP realizing disjunction of attributes. The scheme uses ciphertext-policy attribute-based encryption (CP-ABE) of Ibraimi *et al.* (Ibraimi et al., 2009) and Boneh-Boyan (BB) (Boneh and Boyen, 2004) signature. The CP-ABE of Ibraimi *et al.* (Ibraimi et al., 2009) is not policy hiding. To fit in our construction, we first convert Ibraimi *et al.*'s protocol in to policy hiding CP-ABE. The CP-ABE controls receivers entitled to recover the messages while policy hiding CP-ABE hides the access policies associated with each message together with restrictions on entitled receivers. The BB (Boneh and Boyen, 2004) signature is used to check on the malicious behavior of receivers. It helps one to verify whether the receiver has requested the same ciphertext in transfer phase which was previously published by the sender in initialization phase. The malicious behavior of the sender and receivers is controlled by providing interactive zero-knowledge proofs (Cramer et al., 2000).

The security of the protocol is analyzed in *full-simulation* model following (Camenisch et al., 2012) and (Camenisch et al., 2011) considering the sender's security and the receiver's security separately. In this model, the simulator uses adversarial rewinding to extract the hidden secret in zero-knowledge proofs. Rewinding allows the simulator to rewind the adversary's state to previous computation state and start the computation from there. The proposed protocol is secure under Decision Bilinear Diffie-Hellman (DBDH), q-Strong Diffie-Hellman (SDH) (Boneh and Boyen, 2004) and q-Power Decisional Diffie-Hellman (PDDH) (Camenisch et al., 2007) assumption. The receiver's security is achieved by proving that the sender does not learn who queries a message and which message is being queried. The sender's security is analyzed by proving that a receiver– (i) learns only one message in each transfer phase, (ii) learns nothing about the access policies associated with the messages and (iii) learns only those messages for which it has the secret key associated with attributes that satisfy the access policy of the message.

In contrast to (Camenisch et al., 2012) and (Camenisch et al., 2011) which cover only conjunction of attributes, our scheme realizes disjunction of attributes as well, thereby realizing more expressive access policies as compared to (Camenisch et al., 2012) and (Camenisch et al., 2011). The proposed AOT-HAP protocol outperforms significantly in terms of both computation and communication overheads as compared to (Camenisch et al., 2012) and (Camenisch et al., 2011).

2 PRELIMINARIES

Throughout, we use the notations given in Table 1. A function $f(n)$ is *negligible* if $f = o(n^{-c})$ for every fixed positive constant c.

Table 1: Notations.

Symbol	Description
ρ	Security parameter
$x \xleftarrow{S} X$	Sample x uniformly at random from the set X
$y \leftarrow Y$	y is the output of algorithm Y
DB	Database
AP	Access policy
N	Database size
n	Total number of receivers
m	Total number of attributes
$\mathbb{N}$	Set of natural numbers
$\mathcal{U} = \{1,2,\ldots,n\}$	Universe of receivers
$X \overset{c}{\approx} Y$	Computationally indistinguishable
ID_U	Identity of user $U \in \mathcal{U}$
$\Omega = \{a_1,a_2,\ldots,a_m\}$	Universe of attributes

2.1 Access Structure and Satisfiability

Access structure (Beimel, 1996): Let $\{P_1,P_2,\ldots,P_t\}$ be a set of t parties. A collection $\mathbb{A} \subseteq 2^{\{P_1,P_2,\ldots,P_t\}}$ is monotone if $B \in \mathbb{A}$ and $C \supseteq B$ implies $C \in \mathbb{A}, \forall\, B,C$. An access structure (respectively, monotone access structure) is a collection (respectively, monotone collection) $\mathbb{A}$ of non-empty subsets of $\{P_1,P_2,\ldots,P_t\}$, i.e, $\mathbb{A} \subseteq 2^{\{P_1,P_2,\ldots,P_t\}} \setminus \emptyset$. The sets in $\mathbb{A}$ are called the authorized sets, and the sets not in $\mathbb{A}$ are called the unauthorized sets.

Access tree (Goyal et al., 2006): Let Γ be a tree representing an access structure. Each non-leaf node of the tree represents a threshold gate, described by its children and a threshold value. If num_x is the number of children of a node x and k_x is its threshold value, then $0 < k_x \leq num_x$. The children of a node x are numbered from 1 to num_x. When $k_x = 1$, threshold gate is an OR gate and when $k_x = num_x$, it is an AND gate. Each leaf node x of the tree is described by an attribute and a threshold value $k_x = 1$.

Satisfying an access tree (Goyal et al., 2006): Let Γ be an access tree with root r. Denote by Γ_x the subtree of Γ rooted at the node x. Hence Γ is same as Γ_r. If a set of attributes w satisfies the access tree Γ_x, we denote it by $\Gamma_x(w) = 1$. We compute $\Gamma_x(w)$ recursively as follows. If x is a non-leaf node, evaluate $\Gamma_{x'}(w)$ for all children x' of node x. $\Gamma_x(w)$ returns 1 iff atleast k_x children return 1. If x is a leaf node, then $\Gamma_x(w)$ returns 1 iff $\mathsf{att}(x) \in w$, where the function $\mathsf{att}(x)$ is defined only if x is a leaf node and denotes the attribute associated with the leaf node x in the tree.

2.2 Bilinear Pairing and Complexity Assumptions

Bilinear Pairing: Let $\mathbb{G}_1, \mathbb{G}_2$ and $\mathbb{G}_T$ be three multiplicative cyclic groups of prime order p and g_1 and g_2 be generators of group $\mathbb{G}_1$ and $\mathbb{G}_2$ respectively. Then the map $e : \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T$ is *bilinear* if it satisfies the following conditions:
(i) Bilinear – $e(x^a, y^b) = e(x,y)^{ab} \ \forall \ x \in \mathbb{G}_1, y \in \mathbb{G}_2, a, b \in \mathbb{Z}_p$. (ii) Non-Degenerate – $e(x,y)$ generates $\mathbb{G}_T$, $\forall\, x \in \mathbb{G}_1, y \in \mathbb{G}_2, x \neq 1, y \neq 1$. (iii) Computable – The pairing $e(x,y)$ is computable efficiently $\forall\, x \in \mathbb{G}_1, y \in \mathbb{G}_2$.
If $\mathbb{G}_1 = \mathbb{G}_2$, then e is *symmetric* bilinear pairing. Otherwise, e is *asymmetric* bilinear pairing. Throughout the paper, we use symmetric bilinear pairing.
q-Strong Diffie-Hellman (SDH) assumption (Boneh and Boyen, 2004): Let $\mathbb{G}$ be a multiplicative cyclic group of prime order p with generator g. The q-SDH assumption in $\mathbb{G}$ states: given $(q+1)$-tuple $(g, g^x, g^{x^2}, \ldots, g^{x^q}), x \in \mathbb{Z}_p$, it is hard to find a pair $(c, g^{\frac{1}{x+c}}), c \in \mathbb{Z}_p$.
The q-SDH assumption holds in generic group model (Boneh and Boyen, 2004).
q-Power decisional Diffie-Hellman (PDDH) assumption (Camenisch et al., 2007): Let $\mathbb{G}$ and $\mathbb{G}_T$ be multiplicative cyclic groups of prime order p and $e : \mathbb{G} \times \mathbb{G} \to \mathbb{G}_T$ be a symmetric bilinear pairing. The q-PDDH assumption in $(\mathbb{G}, \mathbb{G}_T)$ states: given $(g, g^x, g^{x^2}, \ldots, g^{x^q}, H), x \in \mathbb{Z}_p, g \xleftarrow{\$} \mathbb{G}^*, H \xleftarrow{\$} \mathbb{G}_T$, it is hard to distinguish the vector $V = (H^x, H^{x^2}, \ldots, H^{x^q}) \in \mathbb{G}_T^q$ from a random vector of $\mathbb{G}_T^q$.
The q-PDDH assumption holds in generic bilinear group model.
Decision bilinear Diffie-Hellman (DBDH) assumption (Ibraimi et al., 2009): Let $\mathbb{G}$ and $\mathbb{G}_T$ be multiplicative cyclic groups of prime order p and $e : \mathbb{G} \times \mathbb{G} \to \mathbb{G}_T$ be a symmetric bilinear pairing. The DBDH assumption in $(\mathbb{G}, \mathbb{G}_T)$ informally states that given $(g, g^a, g^b, g^c), a, b, c \in \mathbb{Z}_p$ as input, it is hard to distinguish $e(g,g)^{abc}$ from a random element $\mathbb{G}_T$.

2.3 Zero-Knowledge Proof of Knowledge

A zero-knowledge proof of knowledge (Bellare and Goldreich, 1993) is a two-party interactive protocol between the prover and the verifier. We use the notation of Camenisch and Stadler (Camenisch and Stadler, 1997) for the various zero-knowledge proofs of knowledge of discrete logarithms and proofs of validity of statements about discrete logarithms. For instance,

$$\mathsf{POK}\{(a,b,c,d) \mid y_1 = g^a h^b \wedge y_2 = g^c h^d\} \quad (1)$$

represents the zero-knowledge proof of knowledge of integers a, b, c and d such that $y_1 = g^a h^b$ and $y_2 = g^c h^d$ holds, where $a, b, c, d \in \mathbb{Z}_p, y_1, y_2, g, h \in \mathbb{G}$, where $\mathbb{G}$ is a cyclic group of prime order p with generator g. The convention is that the quantities in the parenthesis denote elements the knowledge of which are being proved to the verifier by the prover while all other parameters are known to the verifier. The protocol should satisfy two properties. First, it should be *proof of knowledge*, i.e, the prover convinces the verifier that it knows the integers a, b, c and d such that $y_1 = g^a h^b$ and $y_2 = g^c h^d$ hold without revealing anything about the integers a, b, c and d. More technically, a proof is said to be *sound* if there exists a knowledge extractor that extracts the secret values from a successful prover with all but negligible probability. Second, it should be *zero-knowledge*, i.e, the verifier does not learn anything about the integers a, b, c and d. More technically, a proof is said to be *perfect zero-knowledge* if there exists a simulator which without knowing secret values, yields a distribution that cannot be distinguished from the distribution of the transcript generated by the interaction with a real prover. The protocol completes in three rounds. Let us illustrate how the prover and the verifier interact to verify the equation 1. In the first round, the prover picks $z_1, z_2, z_3, z_4 \xleftarrow{\$} \mathbb{Z}_p$, computes $y_3 = g^{z_1} h^{z_2}, y_4 = g^{z_3} h^{z_4}$ and sends y_3, y_4 to the verifier. This round computes four exponentiations in $\mathbb{G}$. In the second round, the verifier chooses a challenge $r \xleftarrow{\$} \mathbb{Z}_p$ and gives it to the prover. In the third round, the prover sets $s_1 = z_1 + r \cdot a, s_2 = z_2 + r \cdot b, s_3 = z_3 + r \cdot c, s_4 = z_4 + r \cdot d$ and sends s_1, s_2, s_3, s_4 to the verifier. The verifier accepts the proof if $g^{s_1} h^{s_2} = y_3 \cdot y_1^r$ and $g^{s_3} h^{s_4} = y_4 \cdot y_2^r$, otherwise, rejects the proof. This round requires six exponentiations in $\mathbb{G}$. The communication complexity is 2 elements from $\mathbb{G}$ and 5 elements from $\mathbb{Z}_p$.

2.4 Formal Model and Security Notions

Communication Model: The adaptive oblivious transfer with hidden access policy (AOT-HAP) is run between a sender S and one or more receivers together with an issuer. The sender S holds a database $\mathsf{DB} = ((m_1, \mathsf{AP}_1), (m_2, \mathsf{AP}_2), \ldots, (m_N, \mathsf{AP}_N))$. Each message m_i in DB is associated with an access policy $\mathsf{AP}_i, i = 1, 2, \ldots, N$. Each receiver R has some attributes. The issuer generates the public key and secret key pair to provide attribute secret keys associated with the attributes of the receivers. The AOT-HAP completes in two phases– *initialization phase* and *transfer phase*. In initialization phase, S encrypts each message m_i of DB associated with AP_i in order to generate ciphertext database $\mathsf{cDB} = (\Phi_1, \Phi_2, \ldots, \Phi_N)$. The sender S does not embed access policies in cDB. In transfer phase, R interacts with S and the issuer and recovers k messages of its choice sequentially. In each transfer phase, R has input $\sigma_j \in [1, N], j = 1, 2, \ldots, k$, and recovers m_{σ_j} after interacting with S and the issuer.

Syntactic of AOT-HAP**:** The AOT-HAP protocol consists of three PPT algorithms Isetup, DBSetup, DBInitialization in addition to two PPT interactive protocols Issue and Transfer which are explained below.

– Isetup**:** The issuer with input security parameter ρ runs this algorithm to generate public parameters params, public key PK_I and secret key SK_I. The issuer publishes params, PK_I and keeps SK_I secret to itself.

– DBSetup**:** This algorithm is run by the sender S who holds the database DB. It generates public and secret key pair $(\mathsf{pk}_{\mathsf{DB}}, \mathsf{sk}_{\mathsf{DB}})$ for S. The sender S publishes public key $\mathsf{pk}_{\mathsf{DB}}$ and keeps secret key $\mathsf{sk}_{\mathsf{DB}}$ secret to itself.

– DBInitialization**:** The sender S with input params, PK_I, $\mathsf{pk}_{\mathsf{DB}}$, $\mathsf{sk}_{\mathsf{DB}}$ and DB runs algorithm DBInitialization, where $\mathsf{DB} = ((m_1, \mathsf{AP}_1), (m_2, \mathsf{AP}_2), \ldots, (m_N, \mathsf{AP}_N))$, AP_i being an access policy for message $m_i, i = 1, 2, \ldots, N$. This algorithms encrypts the database DB in order to generate ciphertext database $\mathsf{cDB} = (\Phi_1, \Phi_2, \ldots, \Phi_N)$, where access policy AP_i is not embedded explicitly in the corresponding ciphertext $\Phi_i, i = 1, 2, \ldots, N$. The sender S publishes cDB and keeps $\mathsf{AP}_1, \mathsf{AP}_2, \ldots, \mathsf{AP}_N$ secret to itself.

– Issue protocol**:** The receiver R with input identity $\mathsf{ID}_R \in \mathcal{U}$ and attribute set $w_{\mathsf{ID}_R} \subseteq \Omega$ interacts with the issuer through a secure communication channel. The issuer uses its public key PK_I and secret key SK_I to generate attribute secret key $\mathsf{ASK}_{w_{\mathsf{ID}_R}}$ for R and sends it in a secure manner to R.

– Transfer protocol**:** The receiver R on input ID_R, index $\sigma \in [1, N]$, ciphertext Φ_σ under access policy AP_σ, $\mathsf{ASK}_{w_{\mathsf{ID}_R}}$ and PK_I interacts with S who holds $(\mathsf{pk}_{\mathsf{DB}}, \mathsf{sk}_{\mathsf{DB}})$ for the database DB, where $\mathsf{ASK}_{w_{\mathsf{ID}_R}}$ is the attribute secret key of R for the attribute set w_{ID_R}. By executing this protocol, R gets m_σ *if* w_{ID_R} *satisfies* AP_σ. Otherwise, R outputs $\perp$.

The access policy in our construction is an access tree in which leaves are attributes and internal nodes are $\wedge$ and $\vee$ boolean operators. The access policy represents which combination of attributes can decrypt the ciphertext. For instance, consider the encryption of a ciphertext Φ with access policy $\mathsf{AP} = a_1 \wedge (a_4 \vee (a_2 \wedge a_3))$, where a_1, a_2, a_3, a_4 are attributes. The set w satisfying this access policy AP is either (a_1, a_4) or (a_1, a_2, a_3) or (a_1, a_2, a_3, a_4). The decrypter can decrypt Φ if it has the attribute secret key ASK_w associated with the attribute set w.

Security Model: The security framework adapted in this paper is in *simulation-based-model* following (Camenisch et al., 2011). This model consists of a *real world* and an *ideal world*. In the real world, parties (a sender, an issuer and one or more receivers) communicate with each other using a real protocol Π. In this world, some of the parties may be corrupted and remain corrupted throughout the execution of the protocol. The corruption is static. Corrupted parties are controlled by the *real world adversary* $\mathcal{A}$. Honest parties follow the protocol Π honestly. In the ideal world, parties and *ideal world adversary* $\mathcal{A}'$ communicate by sending inputs to and receiving outputs from an ideal *functionality* $\mathcal{F}$. All the parties are honest in the ideal world. The *environment machine* $\mathcal{Z}$ which is always activated first is introduced to oversee the execution of $\mathcal{F}$ in the ideal world and the execution of the protocol Π in the real world. It interacts freely with $\mathcal{A}$ throughout the execution of the protocol Π in the real world and with $\mathcal{A}'$ throughout the execution of $\mathcal{F}$ in the ideal world. We describe below how the parties communicate in both the worlds upon receiving messages from $\mathcal{Z}$.

– Real world: The sender and the issuer do not return anything to $\mathcal{Z}$, but the receiver does in the real world.

- The issuer generates the public parameters params, public key PK_I and secret key SK_I by running the algorithm Isetup. It publishes params, PK_I and keeps SK_I secret to itself.
- The sender S runs the algorithm DBSetup in order to generate public key $\mathsf{pk}_{\mathsf{DB}}$ and secret key $\mathsf{sk}_{\mathsf{DB}}$. It publishes $\mathsf{pk}_{\mathsf{DB}}$ and keeps $\mathsf{sk}_{\mathsf{DB}}$ secret to itself.
- The receiver R upon receiving the message (issue, ID_R, w_{ID_R}) from $\mathcal{Z}$ engages in an Issue protocol with the issuer on input ID_R and attribute set w_{ID_R}. After completion of Issue protocol, R returns (is-

sue, ID_R, b) to $\mathcal{Z}$ in response to the message (issue, ID_R, w_{ID_R}), where $b \in \{0,1\}$. The random coin $b = 1$ means that R has obtained the attribute secret key $\mathsf{ASK}_{w_{\mathsf{ID}_R}}$ for attribute set $w_{\mathsf{ID}_R} \subseteq \Omega$. Otherwise, R has failed.

- Upon receiving the message $(\mathsf{encDB}, \mathsf{DB})$, where $\mathsf{DB} = ((m_1, \mathsf{AP}_1), (m_2, \mathsf{AP}_2), \ldots, (m_N, \mathsf{AP}_N))$ from $\mathcal{Z}$, S runs the DBInitialization algorithm to generate ciphertext database $\mathsf{cDB} = (\Phi_1, \Phi_2, \ldots, \Phi_N)$ under their respective access policies $(\mathsf{AP}_1, \mathsf{AP}_2, \ldots, \mathsf{AP}_N)$. The sender S publishes ciphertext database cDB and keeps $\mathsf{AP}_1, \mathsf{AP}_2, \ldots, \mathsf{AP}_N$ secret to itself.
- The receiver R with identity ID_R upon receiving the message (transfer, ID_R, σ) from $\mathcal{Z}$ engages in an Transfer protocol with S. If the transfer succeeded, R returns (transfer, ID_R, m_σ) to $\mathcal{Z}$ in response to the message (transfer, ID_R, σ). Otherwise, R returns (transfer, ID_R, $\perp$) to $\mathcal{Z}$

– Ideal world: All parties communicate through an ideal functionality $\mathcal{F}$ in the ideal world. The honest parties upon receiving the message (issue, ID_R, w_{ID_R}), $(\mathsf{encDB}, \mathsf{DB})$ or (transfer, ID_R, σ) from $\mathcal{Z}$ transfer it to $\mathcal{F}$. We briefly explain the behavior of $\mathcal{F}$. The ideal functionality $\mathcal{F}$ keeps an attribute set w_{ID_R} for each receiver R which is initially set to be empty.

- The ideal functionality $\mathcal{F}$ upon receiving the message (issue, ID_R, w_{ID_R}) from R with identity $\mathsf{ID}_R \in \mathcal{U}$, sends (issue, ID_R, w_{ID_R}) to the issuer. The issuer sends back a bit $c = 1$ to $\mathcal{F}$ in response to the message (issue, ID_R, w_{ID_R}) if the issuer successfully generates the attribute secret key $\mathsf{ASK}_{w_{\mathsf{ID}_R}}$ for a receiver corresponding to its attribute set w_{ID_R}. For $c = 1$, $\mathcal{F}$ sets $w_{\mathsf{ID}_R} = w_{\mathsf{ID}_R}$. Otherwise, $\mathcal{F}$ does nothing.
- Upon receiving the message $(\mathsf{encDB}, \mathsf{DB})$ from the sender S, where $\mathsf{DB} = ((m_1, \mathsf{AP}_1), (m_2, \mathsf{AP}_2), \ldots, (m_N, \mathsf{AP}_N))$, $\mathcal{F}$ records $\mathsf{DB} = ((m_1, \mathsf{AP}_1), (m_2, \mathsf{AP}_2), \ldots, (m_N, \mathsf{AP}_N))$.
- The ideal functionality $\mathcal{F}$ upon receiving the message (transfer, ID_R, σ) from R, checks whether $\mathsf{DB} = \perp$. If $\mathsf{DB} \neq \perp$, $\mathcal{F}$ sends the message (transfer) to S. The sender S sends back a bit d in response to the message (transfer). If the transfer succeeds, S sets $d = 1$. For $d = 1$, $\mathcal{F}$ checks if $\sigma \in [1, n]$ and w_{ID_R} satisfies AP_σ embedded in DB. Then $\mathcal{F}$ sends m_σ to R. Otherwise, it sends $\perp$ to R.

Let $\mathsf{REAL}_{\Pi,\mathcal{Z},\mathcal{A}}$ be the output of $\mathcal{Z}$ after interacting with $\mathcal{A}$ and the parties running the protocol Π in the real world. Also, let $\mathsf{IDEAL}_{\mathcal{F},\mathcal{Z},\mathcal{A}'}$ be the output of $\mathcal{Z}$ after interacting with $\mathcal{A}'$ and parties interacting with $\mathcal{F}$ in the ideal world. The task of $\mathcal{Z}$ is to distinguish with *non-negligible* probability $\mathsf{REAL}_{\Pi,\mathcal{A},\mathcal{Z}}$ from $\mathsf{IDEAL}_{\mathcal{F},\mathcal{A}',\mathcal{Z}}$. The protocol is said to be secure if $\mathsf{REAL}_{\Pi,\mathcal{A},\mathcal{Z}}$ is *computationally indistinguishable* from $\mathsf{IDEAL}_{\mathcal{F},\mathcal{A}',\mathcal{Z}}$.

3 CONCRETE CONSTRUCTION

A high level description of our adaptive oblivious transfer protocol with hidden access policy (AOT-HAP) is as follows. In initialization phase, the sender S with the database $\mathsf{DB} = ((m_1, \mathsf{AP}_1), (m_2, \mathsf{AP}_2), \ldots, (m_N, \mathsf{AP}_N))$ signs the index i of each message m_i with the BB signature to keep a check on the malicious behavior of the receiver R. The signed index i is moved to group $\mathbb{G}_T$ as $e(A_i, h)$, where A_i is the BB signature on index i. The message $m_i \in \mathbb{G}_T$ is masked with signed index i and the component $B_i = e(A_i, h) \cdot m_i$ is encrypted using CP-ABE of (Ibraimi et al., 2009) under the access policy AP_i associated with index i. The CP-ABE of B_i is D_i, where $D_i = (K_i^{(0)}, K_i^{(1)}, K_{i,j}^{(2)})$. The access policy is not made public. The sender S also gives zero-knowledge proof of knowledge of exponents used in generating ciphertext database $\mathsf{cDB} = (\Phi_1, \Phi_2, \ldots, \Phi_N)$. In each transfer phase, whenever R wants to decrypt a ciphertext Φ_{σ_j} with a set of attributes w_{ID_R}, R engages in Issue protocol with the issuer. The issuer generates the attribute secret key $\mathsf{ASK}_{w_{\mathsf{ID}_R}}$ for w_{ID_R} and gives it to R. With $\mathsf{ASK}_{w_{\mathsf{ID}_R}} = (d_0, d_l \ \forall \ a_l \in w_{\mathsf{ID}_R})$, R computes $I_{\sigma_j} = e(K_{\sigma_j}^{(1)}, d_0)$ and $J_{\sigma_j} = \prod_{a_l \in w_{\mathsf{ID}_R}} e\left(K_{\sigma_j,l}^{(2)}, d_l\right)$ and randomizes it. To make sure that R has randomized the ciphertext that was previously published by S, the receiver R proves knowledge of a valid signature for its randomized ciphertext without revealing anything. In order to recover the message m_{σ_j}, R engages in Transfer protocol with S. Formally, our scheme works as follows. To generate bilinear pairing, we invoke algorithm BilinearSetup which on input security parameter ρ generates $\mathsf{params} = (p, \mathbb{G}, \mathbb{G}_T, e, g)$, where $e : \mathbb{G} \times \mathbb{G} \to \mathbb{G}_T$ is a symmetric bilinear pairing, g is a generator of group $\mathbb{G}$ and p, the order of the groups $\mathbb{G}$ and $\mathbb{G}_T$, is prime, i.e $\mathsf{params} = (p, \mathbb{G}, \mathbb{G}_T, e, g) \leftarrow \mathsf{BilinearSetup}(1^\rho)$.

– Isetup: The issuer on input ρ generates $\mathsf{params} = (p, \mathbb{G}, \mathbb{G}_T, e, g) \leftarrow \mathsf{BilinearSetup}(1^\rho)$. It picks $\alpha, t_1, t_2, \ldots, t_m \xleftarrow{\$} \mathbb{Z}_p^*$ and computes $T_j = g^{t_j}, j = 1, 2, \ldots, m, Y = e(g,g)^\alpha$. The public key is $\mathsf{PK}_I = (\mathsf{params}, Y, T_1, T_2, \ldots, T_m)$ and secret key is $\mathsf{SK}_I = (\alpha, t_1, t_2, \ldots, t_m)$. The issuer publishes PK_I to all the parties and keeps SK_I secret to itself.

– DBSetup: The sender S with input params generates

setup parameters for the database DB. It first picks $x, \beta, \gamma \xleftarrow{\$} \mathbb{Z}_p^*$, $h \xleftarrow{\$} \mathbb{G}$ and sets $y = g^x, H = e(g,h), Z = e(g,g)^\beta, P = e(g,g)^\gamma$. The public key is $\mathsf{pk_{DB}} = (H, Z, P, y)$ and secret key is $\mathsf{sk_{DB}} = (h, x, \beta, \gamma)$. The sender S publishes $\mathsf{pk_{DB}}$ to all parties and keeps $\mathsf{sk_{DB}}$ secret to itself. The sender S gives proof of knowledge $\mathsf{POK}\{(h,\beta,\gamma)|H = e(g,h) \wedge Z = e(g,g)^\beta \wedge P = e(g,g)^\gamma\}$ to R. Each receiver R upon receiving $\mathsf{pk_{DB}}$ checks the correctness of $\mathsf{pk_{DB}}$ by verifying the POK. If it fails, R aborts the execution. Otherwise, R accepts $\mathsf{pk_{DB}}$.

– DBInitialization: The sender S on input $\mathsf{PK}_I, \mathsf{params}, \mathsf{pk_{DB}}, \mathsf{sk_{DB}}$ and DB computes ciphertext database $\mathsf{cDB} = (\Phi_1, \Phi_2, \ldots, \Phi_N)$, where $\mathsf{DB} = ((m_1, \mathsf{AP}_1), (m_2, \mathsf{AP}_2), \ldots, (m_N, \mathsf{AP}_N)), m_i \in \mathbb{G}_T, i = 1,2,\ldots,N$. Each message m_i is associated with access policy $\mathsf{AP}_i, i = 1,2,\ldots,N$. The ciphertext Φ_i for each message $m_i, i = 1,2,\ldots,N$, is generated by S as follows.

1. Parse params to extract g and $\mathsf{sk_{DB}}$ to extract x. Generate the BB signature on index i as $A_i = g^{\frac{1}{x+i}}$. The signature is computed to keep an eye on the malicious activities of R. If R deviates from the protocol specification during transfer phase, it will get detected.
2. Compute $B_i = e(A_i, h) \cdot m_i$.
3. In order to hide the access policy AP_i associated with each message m_i, encrypt B_i under the access policy AP_i as explained below.
   (a) Pick $s_i \xleftarrow{\$} \mathbb{Z}_p$ and compute $K_i^{(0)} = B_i \cdot Y^{s_i}$, $K_i^{(1)} = g^{\beta s_i}$, where $Y = e(g,g)^\alpha$ is extracted from PK_I.
   (b) Set the value of root node of access policy AP_i to be s_i. Mark root node assigned and all its child nodes unassigned. Let l be the number of child nodes of root in the access tree corresponding to AP_i. For each unassigned node do the following recursively:
   (i) If the internal node is $\wedge$ and its child nodes are unassigned, assign a value to each unassigned child node by the following technique. For each child node except the last one, assign $r_{i,j} \xleftarrow{\$} \mathbb{Z}_p^*$ and to the last child node assign the value $s_i - \sum_{i=1}^{l-1} r_{i,j}$. Mark these nodes assigned.
   (ii) If the internal node is $\vee$, set the value of each child node to be s_i and mark the node assigned.
   (iii) Let x be a marked node with value $\widetilde{r}$ whose child nodes are yet to be marked. Repeat steps (i) and (ii) by replacing root by node x and value s_i by $\widetilde{r}$.
   (c) For each leaf attribute $a_j \in \mathsf{AP}_i$, compute $K_{i,j}^{(2)} = T_j^{\gamma s_{i,j}}$, where $s_{i,j}$ is the value assigned to leaf node a_j as in step(b). Note that $\sum_{a_j \in w} s_{i,j} = s_i$ for any set of attributes w satisfying the access policy AP_i.
   (d) For $a_j \notin \mathsf{AP}_i$, set $K_{i,j}^{(2)} = T_j^{\gamma s_{i,j}} \cdot g^{z_j}$, $s_{i,j}, z_j \xleftarrow{\$} \mathbb{Z}_p^*$.
   (e) Compute $\pi_i = \mathsf{POK}\{(s_i, s_{i,1}, s_{i,2}, \ldots, s_{i,m}) | Q_i = e(g, K_i^{(1)}) = Z^{s_i} \wedge L_{i,1} = g^{s_{i,1}} \wedge L_{i,2} = g^{s_{i,2}} \wedge \ldots \wedge L_{i,m} = g^{s_{i,m}}\}$.

   The encryption of B_i is $D_i = (K_i^{(0)}, K_i^{(1)}, K_{i,j}^{(2)})$, which is generated following CP-ABE of Ibraimi *et al.* (Ibraimi et al., 2009) together with the zero-knowledge proof of knowledge π_i, $j = 1,2,\ldots,m$.
4. Set $F_i = (Q_i, L_{i,1}, L_{i,2}, \ldots, L_{i,m})$.
5. Set ciphertext $\Phi_i = (A_i, D_i, F_i, \pi_i)$.
6. The ciphertext database $\mathsf{cDB} = (\Phi_1, \Phi_2, \ldots, \Phi_N)$.

The receiver R verifies the proof π_i, and $e(A_i, yg^i) = e(g,g), i = 1,2,\ldots,N$, on receiving ciphertext database cDB. If the verification holds, R accepts cDB. Otherwise, R aborts the execution.

– Issue protocol: The Issue protocol is the interaction between R and the issuer. The input of R is its attribute set w_{ID_R} and identity $\mathsf{ID}_R \in \mathcal{U}$. The issuer picks $r_{\mathsf{ID}_R} \xleftarrow{\$} \mathbb{Z}_p^*$ and sets $d_0 = g^{\alpha - r_{\mathsf{ID}_R}}$, $d_l = g^{r_{\mathsf{ID}_R} \cdot t_l^{-1}} \ \forall \ a_l \in w_{\mathsf{ID}_R}$. The attribute secret key is $\mathsf{ASK}_{w_{\mathsf{ID}_R}} = (d_0, d_l \ \forall \ a_l \in w_{\mathsf{ID}_R})$. The issuer sends $\mathsf{ASK}_{w_{\mathsf{ID}_R}}$ to R through a secure communication channel together with proof of knowledge $\mathsf{POK}\{(\mathsf{SK}_I) | (\mathsf{PK}_I, \mathsf{SK}_I)$ is a key pair$\}$ = $\mathsf{POK}\{(\alpha, t_1, t_2, \ldots, t_m) | Y = e(g,g)^\alpha \wedge T_1 = g^{t_1} \wedge T_2 = g^{t_2} \wedge \ldots \wedge T_m = g^{t_m}\}$ to R. The receiver R verifies the proof. If the verification does not hold, R aborts the execution. Otherwise, R accepts attribute secret key $\mathsf{ASK}_{w_{\mathsf{ID}_R}}$.

– Transfer protocol: The pictorial view of high level description of transfer protocol is given in Figure 1. This protocol is the interaction between S and R. In each of the transfer phase, R picks the index σ_j of its choice with attribute set w_{ID_R}. The receiver R engages in Issue protocol with the issuer in order to obtain the attribute secret key $\mathsf{ASK}_{w_{\mathsf{ID}_R}}$ for the attribute set w_{ID_R}. On receiving $\mathsf{ASK}_{w_{\mathsf{ID}_R}} = (d_0, d_l \ \forall a_l \in w_{\mathsf{ID}_R})$ for w_{ID_R}, R computes I_{σ_j} and J_{σ_j} as follows

$$I_{\sigma_j} = e(K_{\sigma_j}^{(1)}, d_0) = e(g^{\beta s_{\sigma_j}}, g^{\alpha - r_{\mathsf{ID}_R}}),$$
$$J_{\sigma_j} = \prod_{a_l \in w_{\mathsf{ID}_R}} e\left(K_{\sigma_j,l}^{(2)}, d_l\right).$$

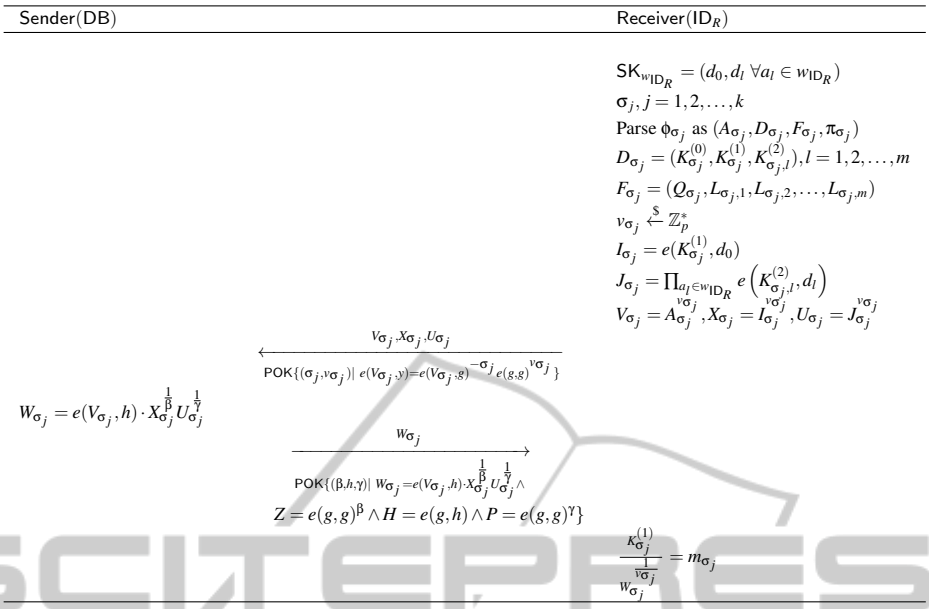

Figure 1: Transfer Protocol.

The receiver R randomizes $A_{\sigma_j}, I_{\sigma_j}, J_{\sigma_j}$ by choosing $v_{\sigma_j} \xleftarrow{\$} \mathbb{Z}_p^*$, sets $V_{\sigma_j} = A_{\sigma_j}^{v_{\sigma_j}}, X_{\sigma_j} = I_{\sigma_j}^{v_{\sigma_j}}$ and $U_{\sigma_j} = J_{\sigma_j}^{v_{\sigma_j}}$ and sends $V_{\sigma_j}, X_{\sigma_j}, U_{\sigma_j}$ to S. The receiver R also gives zero-knowledge proof of knowledge $\mathsf{POK}\{(\sigma_j, v_{\sigma_j})|\ e(V_{\sigma_j}, y) = e(V_{\sigma_j}, g)^{-\sigma_j} e(g,g)^{v_{\sigma_j}}\}$ to S. On verifying the proof, S parses its secret key $\mathsf{sk}_{\mathsf{DB}} = (\beta, x, h, \gamma)$, extracts β, γ and h to generate $W_{\sigma_j} = e(V_{\sigma_j}, h) \cdot X_{\sigma_j}^{\frac{1}{\beta}} U_{\sigma_j}^{\frac{1}{\gamma}}$ and gives it to R together with the zero-knowledge proof of knowledge $\mathsf{POK}\{(\beta, h, \gamma)|\ W_{\sigma_j} = e(V_{\sigma_j}, h) \cdot X_{\sigma_j}^{\frac{1}{\beta}} U_{\sigma_j}^{\frac{1}{\gamma}} \wedge H = e(g,h) \wedge Z = e(g,g)^{\beta} \wedge P = e(g,g)^{\gamma}\}$. The receiver R first verifies the proof and uses its random value v_{σ_j} used to generate $V_{\sigma_j}, I_{\sigma_j}, J_{\sigma_j}$ to recover the message m_{σ_j} as follows

$$\frac{K_{\sigma_j}^{(1)}}{W_{\sigma_j}^{\frac{1}{v_{\sigma_j}}}} = m_{\sigma_j}. \tag{2}$$

The receiver R adaptively runs the transfer phase for k different indexes $\sigma_j, j = 1,2,\ldots,k$. The correctness of equation 2 is given as

$$\begin{aligned}
I_{\sigma_j} &= e(K_{\sigma_j}^{(1)}, d_0) = e(g^{\beta s_{\sigma_j}}, g^{\alpha - r_{\mathsf{ID}_R}}) \\
&= e(g,g)^{\beta s_{\sigma_j}(\alpha - r_{\mathsf{ID}_R})} \\
X_{\sigma_j} &= I_{\sigma_j}^{v_{\sigma_j}} = e(g,g)^{v_{\sigma_j}\beta s_{\sigma_j}(\alpha - r_{\mathsf{ID}_R})} \\
J_{\sigma_j} &= \prod_{a_l \in w_{\mathsf{ID}_R}} e\left(K_{\sigma_j,l}^{(2)}, d_l\right) \\
&= \prod_{a_l \in w_{\mathsf{ID}_R}} e(T_l^{\gamma s_{\sigma_j,l}}, g^{r_{\mathsf{ID}_R} t_l^{-1}}) \\
&= \prod_{a_l \in w_{\mathsf{ID}_R}} e(g^{\gamma t_l s_{\sigma_j,l}}, g^{r_{\mathsf{ID}_R} t_l^{-1}}) \\
&= e(g,g)^{\gamma r_{\mathsf{ID}_R} \sum_{a_l \in w_{\mathsf{ID}_R}} s_{\sigma_j,l}} = e(g,g)^{\gamma r_{\mathsf{ID}_R} s_{\sigma_j}} \\
U_{\sigma_j} &= J_{\sigma_j}^{v_{\sigma_j}} = e(g,g)^{v_{\sigma_j}\gamma r_{\mathsf{ID}_R} s_{\sigma_j}} \\
W_{\sigma_j} &= e(V_{\sigma_j}, h) \cdot X_{\sigma_j}^{\frac{1}{\beta}} U_{\sigma_j}^{\frac{1}{\gamma}} \\
&= \left(e(A_{\sigma_j}, h) e(g,g)^{s_{\sigma_j}(\alpha - r_{\mathsf{ID}_R})} e(g,g)^{r_{\mathsf{ID}_R} s_{\sigma_j}}\right)^{v_{\sigma_j}} \\
&= \left(e(A_{\sigma_j}, h) e(g,g)^{\alpha s_{\sigma_j}}\right)^{v_{\sigma_j}} \\
&= \left(e(A_{\sigma_j}, h) Y^{s_{\sigma_j}}\right)^{v_{\sigma_j}} \text{ as } Y = e(g,g)^{\alpha} \\
\frac{K_{\sigma_j}^{(1)}}{W_{\sigma_j}^{\frac{1}{v_{\sigma_j}}}} &= \frac{e(A_{\sigma_j}, h) m_{\sigma_j} Y^{s_{\sigma_j}}}{W_{\sigma_j}^{\frac{1}{v_{\sigma_j}}}} = m_{\sigma_j}
\end{aligned}$$

Note that $\sum_{a_l \in w_{\mathsf{ID}_R}} s_{\sigma_j,l} = s_{\sigma_j}$ holds only when the attribute set w_{ID_R} satisfies the access policy AP_{σ_j}. Thus although AP_{σ_j} is kept hidden from the receivers, a receiver with a valid attribute set w_{ID_R} (that satisfies AP_{σ_j}) is capable of recovering the message m_{σ_j} encrypted under AP_{σ_j}. A receiver with an attribute set w that does not satisfy AP_{σ_j} will get a random value by decrypting Φ_{σ_j}. For instance, consider the message B_1 with the access policy $\mathsf{AP}_1 = (a_1 \wedge (a_4 \vee (a_2 \wedge a_3)))$, i.e, $\sigma_j = 1$. The CP-ABE of B_1 is as follows. Pick $s_1 \xleftarrow{\$} \mathbb{Z}_p$, set $K_1^{(0)} = B_1 \cdot Y^{s_1}, K_1^{(1)} = g^{\beta s_1}$, $K_{1,1}^{(2)} = T_1^{\gamma s_{1,1}}, K_{1,2}^{(2)} = T_2^{\gamma s_{1,2}}, K_{1,3}^{(2)} = T_3^{\gamma s_{1,3}}, K_{1,4}^{(2)} = T_4^{\gamma s_{1,4}}$ and $K_{1,j}^{(2)} = T_j^{\gamma s_{1,j}} g^{z_j}, s_{1,j}, z_j \xleftarrow{\$} \mathbb{Z}_p, j = 5,6,\ldots,m$. The ciphertext $D_1 = (K_1^{(0)}, K_1^{(1)}, K_{1,j}^{(2)}), j = 1,2,\ldots,m$. The values $s_{1,1}, s_{1,2}, s_{1,3}$ and $s_{1,4}$ used above were generated as follows. The root node of the access policy $\mathsf{AP}_1 = (a_1 \wedge (a_4 \vee (a_2 \wedge a_3)))$ is $\wedge$. Assign value

$s_1 \xleftarrow{\$} \mathbb{Z}_p$ to this node and mark this node assigned. Mark the child nodes unassigned which are a_1 and $\vee$. By the step 3(b)(i) explained in DBInitialization assign value $s_{1,1} = r_{1,1} \xleftarrow{\$} \mathbb{Z}_p$ to a_1 and $s_1 - r_{1,1}$ to $\vee$. Replace the root by node $\vee$ with value $s_1 - r_{1,1}$. By the step 3(b)(ii), assign value $s_{1,4} = s_1 - r_{1,1}$ to a_4 and $s_{1,2} - r_{1,1}$ to $\wedge$. Replace the root by node $\wedge$ with value $s_1 - r_{1,1}$. Following step 3(b)(i), assign value $s_1 \xleftarrow{\$} \mathbb{Z}_p$ to a_2 and $s_1 - r_{1,1} - s_{1,2} = s_{1,3}$ to a_3. Suppose the attribute set $w_1 = \{a_1, a_4\}$ is with a receiver which clearly satisfies the access policy AP_1. Therefore, $\sum_{a_l \in w_1} s_{1,l} = s_{1,1} + s_{1,4} = r_{1,1} + s_1 - r_{1,1} = s_1$.

Note. The CP-ABE scheme of Ibraimi *et al.* (Ibraimi et al., 2009) is not policy hiding, but in our construction we make it policy hiding using secrets β and γ. For instance, consider the encryption of $\mathcal{M}_i$ under the access policy AP_i using CP-ABE of Ibraimi *et al.* (Ibraimi et al., 2009) which is $(K_i^{(0)}, K_i^{(1)}, K_{i,j}^{(2)})$, where

$$\begin{aligned} K_i^{(0)} &= \mathcal{M}_i \cdot Y^{s_i}, \\ K_i^{(1)} &= g^{s_i}, \\ K_{i,j}^{(2)} &= T_j^{s_{i,j}} \quad \text{if } a_j \in \mathsf{AP}_i, \end{aligned}$$

$s_{i,j}$ are taken according to step 3(b) of algorithm DBInitialization. In order to hide the access policy AP_i, we hide $K_i^{(1)}$ using secret β and $K_{i,j}^{(2)}$ using secret γ together with random $K_{i,j}^{(2)}$ for $a_j \notin \mathsf{AP}_i$. Thereby, the encryption of $\mathcal{M}_i$ in our construction is $(K_i^{(0)}, K_i^{(1)}, K_{i,j}^{(2)})$, where

$$\begin{aligned} K_i^{(0)} &= \mathcal{M}_i \cdot Y^{s_i}, \\ K_i^{(1)} &= g^{\beta s_i}, \\ K_{i,j}^{(2)} &= \begin{cases} T_j^{\gamma s_{i,j}}, & a_j \in \mathsf{AP}_i,\ s_{i,j} \text{ as in 3(b)} \\ T_j^{\gamma s_{i,j}} \cdot g^{z_j}, & a_j \notin \mathsf{AP}_i,\ s_{i,j}, z_j \xleftarrow{\$} \mathbb{Z}_p^*. \end{cases} \end{aligned}$$

A receiver is unable to decrypt $\mathcal{M}_i$ using attribute secret key only issued by the issuer because of the secrets β and γ used by the sender during encryption. The receiver has to interact with the sender to recover $\mathcal{M}_i$ correctly. In our construction, $K_{i,j}^{(2)}$ is linear to m whereas in Ibraimi *et al.* $K_{i,j}^{(2)}$ is linear to number of attributes in AP_i. The protocol is constructed in such a way that a receiver will get a correct message only if the receiver's attribute set satisfies the access policy associated with the message implicitly.

4 COMPARISON WITH AOT-HAP IN (Camenisch et al., 2012) AND (Camenisch et al., 2011)

In this section, we compare the proposed scheme with the AOT-HAP in (Camenisch et al., 2012) and (Camenisch et al., 2011) which are the only two AOT-HAP to the best of our knowledge. The proposal of (Camenisch et al., 2011) employed Camenisch *et al.'s* (Camenisch et al., 2007) oblivious transfer, batch Boneh-Boyan (BB) (Boneh and Boyen, 2004) signature and Camenisch-Lysyanskaya (CL) signature (Camenisch and Lysyanskaya, 2004). On the contrary, the AOT-HAP of Camenisch *et al.* (Camenisch et al., 2012) relies on interactive zero-knowledge proofs (Cramer et al., 2000), Groth-Sahai non-interactive proofs (Groth and Sahai, 2008), the privacy friendly signature (Abe et al., 2010) and ciphertext-policy attribute-based encryption (CP-ABE) (Nishide et al., 2008). We point that in (Camenisch et al., 2011), the access policy associated with a message is of the form $\mathsf{AP} = (c_1, c_2, \ldots, c_l)$, where $c_i \in \{0,1\}, i = 1,2,\ldots,l$. On the other hand, the access policy in (Camenisch et al., 2012) is $\mathsf{AP} = (c_1, c_2, \ldots, c_l)$, where $c_i \in [1, n_i], i = 1,2,\ldots,l$. The symbol l denotes the number of categories and n_i is the number of possible attributes for each category. Thus each category c_i in (Camenisch et al., 2012) has n_i values whereas in (Camenisch et al., 2011) each category c_i has only two values. The schemes in (Camenisch et al., 2012) and (Camenisch et al., 2011) covers only conjunction of attributes. In contrast to (Camenisch et al., 2012) and (Camenisch et al., 2011), our scheme employs modified policy hiding Ibraimi *et al.*'s (Ibraimi et al., 2009) CP-ABE and BB (Boneh and Boyen, 2004) signature. Our scheme allows disjunction of attributes as well, thereby realizes more expressive access policy. On a more positive note, the proposed protocol is significantly more efficient as compared to both (Camenisch et al., 2012) and (Camenisch et al., 2011) as illustrated in Tables 2, 3 and 4, where PO stands for the number of pairing, EXP for the number of exponentiation, l denotes the number of categories, m is the total number of attributes, $\alpha X + \beta Y$ represents α elements from the group X and β elements from the group Y. In (Camenisch et al., 2011), $m = 2l$ and in (Camenisch et al., 2012) $m = n_1 + n_2 + \ldots + n_l$. Note that (Camenisch et al., 2012) and (Camenisch et al., 2011) used asymmetric bilinear pairing $e : \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T$. The computation cost also include the cost of verifying the proof of knowledge POK.

Ours Isetup algorithm requires the issuer to compute m EXP in $\mathbb{G}$, 1 EXP in $\mathbb{G}_T$ and 1 PO to generate the public key PK_I, whereas (Camenisch et al., 2012) and

Table 2: Comparison of computation in per ciphertext generation.

AOT-HAP	Sender			Receiver		
	EXP in $\mathbb{G}_1+\mathbb{G}_2$	EXP in $\mathbb{G}_T$	PO	EXP in $\mathbb{G}_1+\mathbb{G}_2$	EXP in $\mathbb{G}_T$	PO
(Camenisch et al., 2012)	$(m+4l+16)\mathbb{G}_1+(4l+10)\mathbb{G}_2$	1	–	$4\mathbb{G}_2$	–	$8l+26$
(Camenisch et al., 2011)	$(4l+7)\mathbb{G}_1+(8l+9)\mathbb{G}_2$	–	1	–	$16l+10$	$144l+19$
Ours	$(3m+2)\mathbb{G}$	3	1	$(2m+1)\mathbb{G}$	2	1

Table 3: Comparison of computation in per Transfer protocol.

AOT-HAP	Sender			Receiver		
	EXP in $\mathbb{G}_1+\mathbb{G}_2$	EXP in $\mathbb{G}_T$	PO	EXP in $\mathbb{G}_1+\mathbb{G}_2$	EXP in $\mathbb{G}_T$	PO
(Camenisch et al., 2012)	$17\mathbb{G}_1+9\mathbb{G}_2$	45	41	$27\mathbb{G}_1+22\mathbb{G}_2$	38	$2l+43$
(Camenisch et al., 2011)	$(12l+104)\mathbb{G}_1$	$4l+51$	$18l+20$	$(16l+99)\mathbb{G}_1+(8l+35)\mathbb{G}_2$	$3l+36$	1
Ours	–	9	4	$1\mathbb{G}$	14	6

(Camenisch et al., 2011) requires $(m+6)$ in $\mathbb{G}_1$, 2 in $\mathbb{G}_2$, 1 in $\mathbb{G}_T$, 3 PO and $(n+3)$ in $\mathbb{G}_1$, 3 in $\mathbb{G}_2$ respectively. Also, the sender computes 1 EXP in $\mathbb{G}$, 2 EXP in $\mathbb{G}_T$ and 1 PO to generate the public key $\mathsf{pk}_{\mathsf{DB}}$ in DBSetup algorithm while that of (Camenisch et al., 2012) and (Camenisch et al., 2011) computes 5 in $\mathbb{G}_1$, 2 in $\mathbb{G}_2$, 2 PO and $l+3$ in $\mathbb{G}_1$, 1 PO respectively.

We emphasize that our scheme computes only a constant number of pairings while that of (Camenisch et al., 2012) and (Camenisch et al., 2011) is linear to l. Total number of exponentiations is less in our scheme as compared to (Camenisch et al., 2012) and (Camenisch et al., 2011). Communication-wise our construction performs favorably over (Camenisch et al., 2012) and (Camenisch et al., 2011). Table 4 compares the communication cost in transferring one ciphertext. It also compares the communication overheads in Transfer protocol. Note that the communication cost also include the cost involved in verifying the proof of knowledge POK.

The communication cost involved in transferring elements the from issuer to R and from R to the issuer is significantly low as compared to (Camenisch et al., 2012) and (Camenisch et al., 2011).

5 SECURITY ANALYSIS

Theorem 1. *The adaptive oblivious transfer with hidden access policy (*AOT-HAP*) decribed in section 2.4 securely implements the* AOT-HAP *functionality assuming the hardness of the q-SDH problem in $\mathbb{G}$, the $(q+1)$-PDDH problem in $\mathbb{G}$ and $\mathbb{G}_T$, the knapsack problem and provided that CP-ABE is fully secure under DBDH assumption, the underlying* POK *is sound and perfect zero-knowledge.*

Proof. The security of the protocol is analyzed by proving indistinguishability between adversary actions in the real protocol and in an ideal scenario. Let $\mathcal{A}$ be a static adversary in the real protocol. We construct an ideal world adversary $\mathcal{A}'$ such that no environment machine $\mathcal{Z}$ can distinguish with non-negligible probability whether it is interacting with $\mathcal{A}$ in the real world or with $\mathcal{A}'$ in the ideal world in the following cases: (a) simulation when only the receiver R is honest, (b) simulation when only the sender S is corrupt (c) simulation when only the receiver R is corrupt (d) simulation when only the sender S is honest. We do not discuss the cases when all the parties (the sender S, the receiver R and the issuer) are honest, when all the parties are corrupt, when only the issuer is honest and when only the issuer is corrupt.

We present the security proof using sequence of hybrid games. Let $\mathsf{Pr}[\mathsf{Game}\ i]$ be the probability that $\mathcal{Z}$ distinguishes the transcript (messages transferred from the sender S to the receiver R and from the receiver R to the sender S) of Game i from the real execution.

(a) Simulation when the sender S and the issuer are corrupt while the receiver R is honest. Firstly, we simulate the interactions of real world. The adversary $\mathcal{A}$ controls the corrupted parties (the sender and the issuer) whereas the simulator simulates the honest receiver R.

Game **0**: The simulator $\mathcal{S}_0$ simulates R and interacts with $\mathcal{A}$ exactly as in the real world. So, $\mathsf{Pr}[\mathsf{Game}\ 0] = 0$. Therefore $\mathsf{REAL}_{\Pi,\mathcal{Z},\mathcal{A}} = \mathsf{Pr}[\mathsf{Game}\ 0]$.

Game **1**: The simulator $\mathcal{S}_1$ works same as $\mathcal{S}_0$ except that $\mathcal{S}_1$ extracts secret key $\mathsf{SK}_I = (\alpha, t_1, t_2, \ldots, t_m)$ by running the knowledge extractor of $\mathsf{POK}\{\mathsf{SK}_I | (\mathsf{PK}_I, \mathsf{SK}_I)$ is a key pair$\}$ when the issue query is instructed by $\mathcal{Z}$. The difference between Game 1 and Game 0 is given by the knowledge error of POK which is negligible provided the underlying POK is sound. Therefore, there exists a negligible function $\varepsilon_1(\rho)$ such that $|\mathsf{Pr}[\mathsf{Game}\ 1] - \mathsf{Pr}[\mathsf{Game}\ 0]| \leq \varepsilon_1(\rho)$.

Game **2**: This game is the same as Game 1 except that the simulator $\mathcal{S}_2$ runs the knowledge extractor of $\mathsf{POK}\{(h, \beta, \gamma) | H = e(g,h) \wedge Z = e(g,g)^{\beta} \wedge P = e(g,g)^{\gamma}\}$ to extract h, β, γ from $\mathcal{A}$. The difference be-

Table 4: Comparison in terms of communication.

AOT-HAP	Per Ciphertext			Per Transfer Protocol		
	$\mathbb{Z}_p$	$\mathbb{G}_1+\mathbb{G}_2$	$\mathbb{G}_T$	$\mathbb{Z}_p$	$\mathbb{G}_1+\mathbb{G}_2$	$\mathbb{G}_T$
(Camenisch et al., 2012)	1	$(m+2l+11)\mathbb{G}_1+(2l+4)\mathbb{G}_2$	1	19	$18\mathbb{G}_1+14\mathbb{G}_2$	21
(Camenisch et al., 2011)	–	$(4l+2)\mathbb{G}_1+(4l+5)\mathbb{G}_2$	1	$(6l+55)\mathbb{G}_1+(4l+40)\mathbb{G}_2$	$4l+7$	17
Ours	$2m+2$	$3m+2$	3	8	2	8

tween Game 2 and Game 1 is the knowledge error of POK which is negligible provided the underlying POK is sound. Therefore, there exists a negligible function $\varepsilon_2(\rho)$ such that $|\Pr[\text{Game 2}]-\Pr[\text{Game 1}]| \leq \varepsilon_2(\rho)$.

Game **3**: The simulator $\mathcal{S}_3$ works same as $\mathcal{S}_2$ except that $\mathcal{S}_3$ extracts the secret exponents $s_i, s_{i,1}, s_{i,2}, \ldots, s_{i,m}$ by running the knowledge extractor of $\pi_i = \mathsf{POK}\{(s_i, s_{i,1}, s_{i,2}, \ldots, s_{i,m}) | Q_i = Z^{s_i} \wedge L_{i,1} = g^{s_{i,1}} \wedge L_{i,2} = g^{s_{i,2}} \wedge \ldots \wedge L_{i,m} = g^{s_{i,m}}\}$ when the sender S publishes ciphertext database cDB upon instructed by $\mathcal{Z}$. The difference between Game 3 and Game 2 is given by the knowledge error of POK which is negligible provided the underlying POK is sound. Therefore, there exists a negligible function $\varepsilon_3(\rho)$ such that $|\Pr[\text{Game 3}]-\Pr[\text{Game 2}]| \leq \varepsilon_3(\rho)$.

Game **4**: The simulator $\mathcal{S}_4$ works same as $\mathcal{S}_3$ except that $\mathcal{S}_4$ engages in a transfer protocol with $\mathcal{A}$ to learn message randomly chosen from those for which $\mathcal{S}_4$ has the necessary decryption key. The difference between Game 4 and Game 3 is negligible due to the perfect zero-knowledgeness of the underlying $\mathsf{POK}\{(\sigma_j, v_{\sigma_j}) | \ e(V_{\sigma_j}, y) = e(V_{\sigma_j}, g)^{-\sigma_j} e(g,g)^{v_{\sigma_j}}\}$. Therefore, there exists a negligible function $\varepsilon_4(\rho)$ such that $|\Pr[\text{Game 4}]-\Pr[\text{Game 3}]| \leq \varepsilon_4(\rho)$.

Now we construct the ideal world adversary $\mathcal{A}'$ with black box access to $\mathcal{A}$. The adversary $\mathcal{A}'$ incorporates all steps from Game 4. The adversary $\mathcal{A}'$ first interacts with $\mathcal{A}$ to get Φ_i, where $\Phi_i = (A_i, D_i, F_i, \pi_i), A_i = g^{\frac{1}{x+i}}, D_i = (K_i^{(0)} = B_i \cdot Y^{s_i}, K_i^{(1)} = g^{\beta s_i}, K_{i,l}^{(2)}), B_i = e(A_i, g) \cdot m_i, \pi_i = \mathsf{POK}\{(s_i, s_{i,1}, s_{i,2}, \ldots, s_{i,m}) | Q_i = e(g, K_i^{(1)}) = Z^{s_i} \wedge L_{i,1} = g^{s_{i,1}} \wedge L_{i,2} = g^{s_{i,2}} \wedge \ldots \wedge L_{i,m} = g^{s_{i,m}}\}, i = 1,2,\ldots,N, l = 1,2,\ldots,m$. The adversary $\mathcal{A}'$ simulates the interactions of R with $\mathcal{A}$ for issuing decryption key. If the decryption key is valid, $\mathcal{A}'$ sends a bit $b=1$ to $\mathcal{F}$, otherwise, it sends $b=0$. If $\mathcal{A}'$ interacts with $\mathcal{A}$ in issue protocol, $\mathcal{A}'$ extracts the secret key $\mathsf{SK}_I = (\alpha, t_1, t_2, \ldots, t_m)$ by the running the knowledge extractor of $\mathsf{POK}\{\mathsf{SK}_I | (\mathsf{PK}_I, \mathsf{SK}_I)$ is a key pair$\}$. Upon receiving the transfer query from $\mathcal{F}$, $\mathcal{A}'$ will query a message randomly chosen from those for which $\mathcal{A}'$ has the necessary decryption key. If the transfer protocol succeeds, $\mathcal{A}'$ sends a bit $b=1$ to $\mathcal{F}$, otherwise, it sends $b=0$. Also $\mathcal{A}'$ runs the knowledge extractor of $\mathsf{POK}\{(\beta, h, \gamma) | \ Z = e(g,g)^{\beta} \wedge H = e(g,h) \wedge P = e(g,g)^{\gamma}\}$ to extract β, h, γ from $\mathcal{A}$. Now $\mathcal{A}'$ parses SK_I to get α and computes $\frac{K_i^{(0)}}{e(A_i,h)e(K_i^{(1)}, g)^{\frac{\alpha}{\beta}}} = m_i$ as $K_i^{(0)} = e(A_i, h) \cdot m_i \cdot Y^{s_i}, K_i^{(1)} = g^{\beta s_i}, Y = e(g,g)^{\alpha}$. The adversary $\mathcal{A}'$ extracts attributes associated with m_i as follows. Let atr_i be the set of attributes associated with m_i which is initially set to be empty. The adversary $\mathcal{A}'$ parses F_i as $(Q_i, L_{i,1}, L_{i,2}, \ldots, L_{i,m})$ and checks if $K_{i,l}^{(2)} = (L_{i,l})^{\eta_l}$, where t_l is extracted from SK_I. If so, then $\mathsf{atr}_i = \mathsf{atr}_i \cup \{a_l\}, l = 1,2,\ldots,m$. In this way, $\mathcal{A}'$ obtains the attribute set atr_i associated with message m_i. The adversary $\mathcal{A}'$ constructs AP_i by finding all possible solutions of

$$\prod_{a_t \in \mathsf{atr}_i} (L_{i,t})^{x_t} = (K_i^{(1)})^{\frac{1}{\beta}}, x_t \in \{0,1\}. \qquad (3)$$

Note that the equation 3 can be viewed as an instance of the knapsack problem as finding a solution of the equation 3 is essentially the same as finding solution of $\sum_{t \in I_i} s_{i,t} x_t = s_i$, where $x_t \in \{0,1\}$, $I_i = \{t \mid a_t \in \mathsf{atr}_i\}$ and $s_{i,t}, s_i$ are extracted by $\mathcal{A}'$ by running the knowledge extractor of π_i embedded in Φ_i. A subset of atr_i for which the equation 3 holds is a clause of AP_i and disjuncting all these clauses provides the required access policy AP_i, where $i = 1,2,\ldots,N$. The adversary $\mathcal{A}'$ sends $((m_1, \mathsf{AP}_1), (m_2, \mathsf{AP}_2), \ldots, (m_N, \mathsf{AP}_N))$ to $\mathcal{F}$ for *encDB*. We note that $\mathcal{A}'$ provides $\mathcal{A}$ the same environment as simulator $\mathcal{S}_4$ provided $\mathcal{A}'$ can solve the knapsack problem with negligible error. So, we have $\mathsf{IDEAL}_{\mathcal{F},\mathcal{Z},\mathcal{A}'} = \Pr[\text{Game 4}] + \varepsilon_{knap}$ and $\mathsf{IDEAL}_{\mathcal{F},\mathcal{Z},\mathcal{A}'} - \mathsf{REAL}_{\Pi,\mathcal{Z},\mathcal{A}} = |\Pr[\text{Game 4}] - [\text{Game 0}]| + \varepsilon_{knap} \leq |\Pr[\text{Game 4}] - [\text{Game 3}]| + |\Pr[\text{Game 3}] - [\text{Game 2}]| + |\Pr[\text{Game 2}] - [\text{Game 1}]| + |\Pr[\text{Game 1}] - [\text{Game 0}]| + \varepsilon_{knap} \leq \varepsilon_4(\rho) + \varepsilon_3(\rho) + \varepsilon_2(\rho) + \varepsilon_1(\rho) + \varepsilon_{knap} = \nu(\rho)$, where $\nu(\rho)$ and ε_{knap} are negligible functions. Hence $\mathsf{IDEAL}_{\mathcal{F},\mathcal{Z},\mathcal{A}'} \overset{c}{\approx} \mathsf{REAL}_{\Pi,\mathcal{Z},\mathcal{A}}$.

(b) Simulation when the sender S is corrupt while the receiver R and the issuer are honest. In this case the adversary $\mathcal{A}$ controls the corrupted sender S whereas the simulator simulates the honest receiver R and honest issuer. The simulation of this case is exactly the same as Case(a) except that the simulator itself generates the setup parameters on behalf of the issuer, thereby knows the secret key SK_I which the simulator has to extract in the above case.

(c) Simulation when the sender S and the issuer

are honest while the receiver R is corrupt. In this case, the adversary $\mathcal{A}$ controls the corrupted receiver R and the simulator simulates the honest sender S and honest issuer.

Game **0**: This game corresponds to the real world protocol interaction in which the simulator $\mathcal{S}_0$ simulates S and honest issuer. So, $\mathsf{Pr}[\text{Game 0}]$ =0. Therefore $\mathsf{REAL}_{\Pi,\mathcal{Z},\mathcal{A}} = \mathsf{Pr}[\text{Game 0}]$.

Game **1**: This game is same as Game 0 except that the simulator $\mathcal{S}_1$ extracts (σ_j, v_{σ_j}) by running the knowledge extractor of $\mathsf{POK}\{(\sigma_j, v_{\sigma_j})|\ e(V_{\sigma_j}, y) = e(V_{\sigma_j}, g)^{-\sigma_j} e(g,g)^{v_{\sigma_j}}\}$ for each transfer phase $j, j = 1,2,\ldots,k$. The difference between Game 1 and Game 0 is the knowledge error of POK which is negligible under soundness of the underlying POK. Therefore, there exists a negligible function $\varepsilon_1(\rho)$ such that $|\mathsf{Pr}[\text{Game 1}] - \mathsf{Pr}[\text{Game 0}]| \leq \varepsilon_1(\rho)$.

Game **2**: In this game, the simulator $\mathcal{S}_2$ computes $\widehat{A_{\sigma_j}} = V_{\sigma_j}^{\frac{1}{v_{\sigma_j}}}$ and $e(g, \widehat{K_{\sigma_j}^{(1)}})^{\frac{1}{\beta}} = X_{\sigma_j}^{\frac{1}{\beta v_{\sigma_j}}} U_{\sigma_j}^{\frac{1}{\gamma v_{\sigma_j}}}$ by using v_{σ_j} which is extracted in Game 1. If the adversary $\mathcal{A}$ has never requested the issuer for decryption key for the attribute set w_{ID_R}, then $e(g, \widehat{K_{\sigma_j}^{(1)}})^{\frac{1}{\beta}} = e(g, K_{\sigma_j}^{(1)})^{\frac{1}{\beta}}$ with negligible probability because we can construct an adversary $\mathcal{B}$ to break the security of CP-ABE with black box access to $\mathcal{A}$. Also, if the extracted index $\sigma_j \notin \{1,2,\ldots,N\}$, then one can note that $\widehat{A_{\sigma_j}}$ is a forged BB signature on σ_j. This in turn indicates that $\mathcal{A}$ is able to come up with a valid BB signature $\widehat{A_{\sigma_j}}$, thereby $\mathcal{A}$ outputs $\widehat{A_{\sigma_j}}$ as a forgery contradicting the fact that the BB signature is unforgeable under chosen-message attack assuming q-SDH problem is hard (Boneh and Boyen, 2004).

Hence, there exists a negligible function $\varepsilon_2(\rho)$ such that $|\mathsf{Pr}[\text{Game 2}] - \mathsf{Pr}[\text{Game 1}]| \leq \varepsilon_2(\rho)$.

Game **3**: This game is the same as Game 2 except that the simulator $\mathcal{S}_3$ simulates the response W_{σ_j} as $\left(\frac{K_{\sigma_j}^{(1)}}{m_{\sigma_j}}\right)^{v_{\sigma_j}}$ and also simulates $\mathsf{POK}\{(\beta,h,\gamma)|\ W_{\sigma_j} = e(V_{\sigma_j},h) \cdot X_{\sigma_j}^{\frac{1}{\beta}} U_{\sigma_j}^{\frac{1}{\gamma}} \wedge H = e(g,h) \wedge Z = e(g,g)^{\beta} \wedge P = e(g,g)^{\gamma}\}$. The difference between Game 3 and Game 2 is negligible provided the underlying POK has zero-knowledgeness. Therefore, there exists a negligible function $\varepsilon_3(\rho)$ such that $|\mathsf{Pr}[\text{Game 3}] - \mathsf{Pr}[\text{Game 2}]| \leq \varepsilon_3(\rho)$.

Game **4**: In this game, the simulator $\mathcal{S}_4$ replaces $K_{\sigma_j}^{(1)}$ by random elements of $\mathbb{G}_T$.

Claim 1. The difference between Game 4 *and* Game 3 *is negligible provided that the q-PDDH assumption holds.*

If the environment machine $\mathcal{Z}$ can distinguish between Game 4 and Game 3, we can construct a solver $\mathcal{B}$ for q-PDDH assumption. The adversary $\mathcal{B}$ is given an instance $g', g'^{x}, g'^{x^2}, \ldots, g'^{x^{(q)}}, H', H'_1, \ldots, H'_q, g' \xleftarrow{\$} \mathbb{G}, H' \xleftarrow{\$} \mathbb{G}_T, x \xleftarrow{\$} \mathbb{Z}_p$. The task of $\mathcal{B}$ is to decide whether $H'_l = H'^{x^l}$ or H'_l are just random elements of $\mathbf{G}_T, l = 1,2,\ldots,q$. The adversary $\mathcal{B}$ plays the role of honest sender S and issuer. The adversary $\mathcal{B}$ uses $\mathcal{Z}$ and $\mathcal{A}$ as subroutines. Let $f(x) = \prod_{i=1}^{q}(x+i) = \sum_{i=0}^{q} b_i x^i$ be a polynomial of degree q, where b_i are coefficients of $f(x)$. Set $g = g'^{f(x)} = \prod_{i=0}^{q}(g'^{x^i})^{b_i}$, $H = H'^{f(x)}$ and $y = g'^{xf(x)} = \prod_{i=0}^{q}(g'^{x^{i+1}})^{b_i}$, $Z = e(g,g)^{\beta}, P = e(g,g)^{\gamma}$. The adversary $\mathcal{B}$ sets $\mathsf{pk}_{\mathsf{DB}} = (H, y, Z, P)$. Let $f_i(x) = \frac{f(x)}{x+i} = \sum_{l=0}^{q-1} b_{i,l} x^l$ be a polynomial of degree $q-1$. The adversary $\mathcal{B}$ can compute $A_i = g^{\frac{1}{x+i}} = g'^{\frac{f(x)}{x+i}} = g'^{f_i(x)} = \prod_{l=0}^{q-1}(g'^{x^l})^{b_{i,l}}$ and $K_i^{(1)} = H^{\frac{1}{x+i}} m'_i = H'^{f_i(x)} m'_i = \prod_{l=0}^{q-1}(H'_l)^{b_{i,l}} m'_i$, where $m'_i = m_i \cdot Y^{s_i}$. If $H'_l = (H')^{x^l}$, $\mathcal{B}$ plays the role of simulator $\mathcal{S}_3$ as in Game 3, otherwise, if H'_l are random elements of $\mathbf{G}_T$, $\mathcal{B}$ plays the role of simulator $\mathcal{S}_4$ as in Game 4. Thus if $\mathcal{Z}$ can distinguish between Game 4 and Game 3, $\mathcal{B}$ can solve q-PDDH assumption.

Therefore, by *Claim* 1 there exists a negligible function $\varepsilon_4(\rho)$ such that $|\mathsf{Pr}[\text{Game 4}]\text{-}\mathsf{Pr}[\text{Game 3}]| \leq \varepsilon_4(\rho)$.

We now construct the ideal world adversary $\mathcal{A}'$ with black box access to $\mathcal{A}$. The adversary $\mathcal{A}'$ incorporates all steps from Game 4. The adversary $\mathcal{A}'$ simultaneously plays the role of honest sender S and honest issuer. The adversary $\mathcal{A}'$ sets up $\mathsf{pk}_{\mathsf{DB}}$ on behalf of S and PK_I on behalf of the honest issuer. The adversary $\mathcal{A}'$ generates the ciphertext $\Phi_i = (A_i, D_i, F_i, \pi_i)$ by randomly picking $K_i^{(1)}$ from $\mathbf{G}_T, i = 1,2,\ldots,N$.

Upon receiving the message (issue, ID_R, w_{ID_R}) from $\mathcal{Z}$, $\mathcal{A}'$ sends the message to $\mathcal{A}$. If $\mathcal{A}$ deviates from protocol specification, $\mathcal{A}'$ outputs $\perp$. Otherwise, $\mathcal{A}$ requests $\mathcal{A}'$ for decryption key with attribute set w_{ID_R}. The adversary $\mathcal{A}'$ requests $\mathcal{F}$ for the decryption key. If $\mathcal{F}$ sends the bit $b = 1$ to $\mathcal{A}'$, $\mathcal{A}'$ sends the decryption key to $\mathcal{A}$. Otherwise, $\mathcal{A}'$ sends $b = 0$ to $\mathcal{A}$.

Upon receiving the message (transfer, ID_R, σ_j) from $\mathcal{Z}$, $\mathcal{A}'$ sends the message to $\mathcal{A}$. If $\mathcal{A}$ deviates from protocol specification, $\mathcal{A}'$ outputs $\perp$. Otherwise, $\mathcal{A}'$ extracts (σ_j, v_{σ_j}) from the proof of knowledge given by $\mathcal{A}$. The adversary $\mathcal{A}'$ requests decryption key and message m_{σ_j} from $\mathcal{F}$. The adversary $\mathcal{A}'$ simulates the response W_{σ_j} as $\left(\frac{K_{\sigma_j}^{(1)}}{m_{\sigma_j}}\right)^{v_{\sigma_j}}$ and sends W_{σ_j} along with the simulated zero-knowledge proof to $\mathcal{A}$. Thus the simulation provided by $\mathcal{A}'$ to $\mathcal{A}$ is same as the simulator $\mathcal{S}_4$ as in Game 4. So, we have $\mathsf{IDEAL}_{\mathcal{F},\mathcal{Z},\mathcal{A}'} = \mathsf{Pr}[\text{Game 4}]$ and $\mathsf{IDEAL}_{\mathcal{F},\mathcal{Z},\mathcal{A}'} -$

$\mathsf{REAL}_{\Pi,\mathcal{Z},\mathcal{A}} = |\Pr[\text{Game 4}] - [\text{Game 0}]| \leq |\Pr[\text{Game 4}] - [\text{Game 3}]| + |\Pr[\text{Game 3}] - [\text{Game 2}]||\Pr[\text{Game 2}] - [\text{Game 1}]| + |\Pr[\text{Game 1}] - [\text{Game 0}]| \leq \varepsilon_4(\rho) + \varepsilon_3(\rho) + \varepsilon_2(\rho) + \varepsilon_1(\rho) = \nu(\rho)$, where $\nu(\rho)$ is a negligible function. Hence $\mathsf{IDEAL}_{\mathcal{F},\mathcal{Z},\mathcal{A}'} \overset{c}{\approx} \mathsf{REAL}_{\Pi,\mathcal{Z},\mathcal{A}}$.

(d) Simulation when the sender *S* is honest while the issuer and the receiver *R* are corrupt. In this case the adversary $\mathcal{A}$ controls the corrupted receiver *R* and issuer and simulator simulates the honest sender *S*. The simulation of this case is exactly the same as Case(c) except that in this case the corrupted receivers can obtain all the attribute secret keys they want as the issuer is controlled by the adversary.

□

6 CONCLUSION

We have proposed a scheme in which the sender has published encrypted messages which are protected by hidden access policies. The receiver recovers the message without revealing its identity and choice of message to the sender. The scheme has covered disjunction of attributes. Our construction uses ciphertext policy attribute based encryption and Boneh-Boyan signature. The proposed scheme is secure in the presence of malicious adversary under the q-Strong Diffie-Hellman (SDH) assumption, q-Power Decisional Diffie-Hellman (PDDH) assumption and Decision Bilinear Diffie-Hellman (DBDH) assumption in full-simulation security model. Our scheme is computationally efficient and has low communication overhead.

REFERENCES

Abe, M., Fuchsbauer, G., Groth, J., Haralambiev, K., and Ohkubo, M. (2010). Structure-preserving signatures and commitments to group elements. In *CRYPTO 2010*, volume 6223 of *LNCS*, pages 209–236. Springer.

Beimel, A. (1996). *Secure schemes for secret sharing and key distribution*. PhD thesis, PhD thesis, Israel Institute of Technology, Technion, Haifa, Israel.

Bellare, M. and Goldreich, O. (1993). On defining proofs of knowledge. In *CRYPTO 1992*, volume 740 of *LNCS*, pages 390–420. Springer.

Boneh, D. and Boyen, X. (2004). Short signatures without random oracles. In *EUROCRYPT 2004*, LNCS, pages 56–73. Springer.

Camenisch, J., Dubovitskaya, M., Enderlein, R. R., and Neven, G. (2012). Oblivious transfer with hidden access control from attribute-based encryption. In *SCN 2012*, volume 7485 of *LNCS*, pages 559–579. Springer.

Camenisch, J., Dubovitskaya, M., and Neven, G. (2009). Oblivious transfer with access control. In *ACM 2009*, pages 131–140. ACM.

Camenisch, J., Dubovitskaya, M., Neven, G., and Zaverucha, G. M. (2011). Oblivious transfer with hidden access control policies. In *PKC 2011*, volume 6571 of *LNCS*, pages 192–209. Springer.

Camenisch, J. and Lysyanskaya, A. (2004). Signature schemes and anonymous credentials from bilinear maps. In *CRYPTO 2004*, volume 3152 of *LNCS*, pages 56–72. Springer.

Camenisch, J., Neven, G., et al. (2007). Simulatable adaptive oblivious transfer. In *EUROCRYPT 2007*, volume 4515 of *LNCS*, pages 573–590. Springer.

Camenisch, J. and Stadler, M. (1997). Efficient group signature schemes for large groups. In *CRYPTO 1997*, volume 1294 of *LNCS*, pages 410–424. Springer.

Coull, S., Green, M., and Hohenberger, S. (2009). Controlling access to an oblivious database using stateful anonymous credentials. In *PKC 2009*, volume 5443 of *LNCS*, pages 501–520. Springer.

Cramer, R., Damgård, I., and MacKenzie, P. (2000). Efficient zero-knowledge proofs of knowledge without intractability assumptions. In *PKC 2000*, volume 1751 of *LNCS*, pages 354–372. Springer.

Goyal, V., Pandey, O., Sahai, A., and Waters, B. (2006). Attribute-based encryption for fine-grained access control of encrypted data. In *ACM 2006*, pages 89–98. ACM.

Green, M. and Hohenberger, S. (2007). Blind identity-based encryption and simulatable oblivious transfer. In *ASIACRYPT 2007*, volume 4833 of *LNCS*, pages 265–282. Springer.

Groth, J. and Sahai, A. (2008). Efficient non-interactive proof systems for bilinear groups. In *EUROCRYPT 2008*, volume 4965 of *LNCS*, pages 415–432. Springer.

Ibraimi, L., Tang, Q., Hartel, P., and Jonker, W. (2009). Efficient and provable secure ciphertext-policy attribute-based encryption schemes. In *ISPEC 2009*, volume 5451 of *LNCS*, pages 1–12. Springer.

Naor, M. and Pinkas, B. (1999). Oblivious transfer with adaptive queries. In *CRYPTO 1999*, volume 1666 of *LNCS*, pages 573–590. Springer.

Nishide, T., Yoneyama, K., and Ohta, K. (2008). Attribute-based encryption with partially hidden encryptor-specified access structures. In *ACNS 2008*, volume 5037 of *LNCS*, pages 111–129. Springer.

Zhang, Y., Au, M. H., Wong, D. S., Huang, Q., Mamoulis, N., Cheung, D. W., and Yiu, S.-M. (2010). Oblivious transfer with access control: realizing disjunction without duplication. In *Pairing 2010*, volume 6487 of *LNCS*, pages 96–115. Springer.