Adaptive Oblivious Transfer Realizing Expressive Hidden Access Policy

Vandana Guleria(✉) and Ratna Dutta

Department of Mathematics,
Indian Institute of Technology Kharagpur, Kharagpur 721302, India
vandana.math@gmail.com, ratna@maths.iitkgp.ernet.in

Abstract. *Adaptive Oblivious Transfer Protocol with Hidden Access Policy* (AOT-HAP) is a well known cryptographic primitive that combines each message of a database with an access policy which is kept hidden. The database is held by a sender who encrypts each message of the database under its access policy and publishes encrypted database in which access policies are embedded implicitly. A receiver possesses an attribute set and recovers the message correctly if the attribute set satisfies the access policy implicitly. Otherwise, a garbage message is recovered by the receiver. In this paper, an efficient AOT-HAP is presented. The proposed protocol realizes more expressive access policies, i.e., *conjunction* as well as *disjunction* of attributes. The proposed AOT-HAP is secure assuming the hardness of standard assumptions in the presence of malicious adversary in full-simulation security model. It exhibits significant improvement over the existing similar schemes in terms of both communication and computation.

Keywords: Oblivious transfer · Access policy · Attribute based encryption · Full simulation security model

1 Introduction

Adaptive Oblivious Transfer (AOT) is an interesting area of research nowadays. AOT involves two parties – a sender and a receiver. The sender holds a database of N secret messages, and the receiver wants to get k of them without disclosing which k of them. The protocol completes in one initialization phase and k transfer phases. In initialization phase, the sender encrypts the database of secret messages and publishes the encrypted database for everyone. In each transfer phase, the receiver interacts with the sender to get secret messages. The receiver retrieves k secret messages adaptively, i.e., one in each transfer phase and oblivious to other $N-k$ secret messages. AOT is useful in adaptive oblivious search of large databases such as medical, finance, patent etc.

Sometimes, the sender wants secret messages to be accessed only by selected recipients. For this, each secret message is associated with an access policy, which could be attributes, roles, or rights. The access policy represents which

M.S. Obaidat and A. Holzinger (Eds.): ICETE 2014, CCIS 554, pp. 212–233, 2015.
DOI: 10.1007/978-3-319-25915-4_12

combination of attributes a receiver should have in order to access the secret message. For instance, consider "ms.pdf" file can be downloaded by "CS students of IIT Kharagpur". Here, "CS students of IIT Kharagpur" is the access policy associated with the file "ms.pdf". In some practical scenario, the associated access policy discloses too much information about the secret message. To overcome this, access policies are also kept hidden by the sender. Such primitives are called adaptive oblivious transfer with hidden access policy (AOT-HAP).

The AOT-HAP is executed between a sender, an issuer and a set of receivers. It competes in one initialization phase, one issue phase and k transfer phases. Initialization and issue phases are off-line, whereas, transfer phases are on-line. The sender has a database $\mathsf{DB} = \{(m_i, \mathsf{AP}_i)\}_{1 \leq i \leq N}$ of N of messages, where AP_i is the access policy attached with m_i, $i = 1, 2, \ldots, N$. Each receiver has an attribute set w and interacts with the issuer in issue phase to obtain an attribute secret key for w. The ciphertext database $\mathsf{cDB} = \{\Phi_i\}_{1 \leq i \leq N}$ is made public in initialization phase, where Φ_i is the encryption of m_i under AP_i, $i = 1, 2, \ldots, N$. The protocol is constructed in such a way that the receiver correctly decrypts the ciphertext Φ_{σ_j} to get m_{σ_j} in j-th transfer phase only if w satisfies AP_{σ_j} implicitly, otherwise, a garbage message is retrieved by the receiver, where $\sigma_j \in \{1, 2, \ldots, N\}$, $j = 1, 2, \ldots, k$. 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.

Rabin [22] presented the first oblivious transfer protocol, which was later generalized in [4]. Afterwards, many researchers [9,13,14,17–19,21] proposed AOT protocols. Aforesaid, AOT protocols do not consider access policies. The first AOT with access policy was introduced by Coull *et al.* [11] assuming access policies as state graphs. Later, Camenisch *et al.* [6] introduced AOT with access policy in which access policies were conjunction of attributes (e.g. $a_1 \wedge a_2$, where a_1 and a_2 are attributes). Furthermore, if m is a message with access policy $(a_1 \wedge a_2) \vee (a_3 \wedge a_4)$ in [6], then m is 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. Encryption of the same message multiple times under different access policies is called duplication of the message. This limitation has been eliminated in [23]. Although, access policies are attached with the databases in [6,11,23], but they are not hidden. Recently, [5,7] introduced AOT-HAP which are to the best of our knowledge the only oblivious transfer protocols with hidden access policies.

Our Contribution. Our main focus in this paper is to design an efficient AOT-HAP which cover both conjunction as well as disjunction of attributes. The proposed AOT-HAP is the *first* AOT-HAP realizing disjunction of attributes, to the best of our knowledge. To this end, ciphertext-policy attribute-based encryption (CP-ABE) of Ibraimi *et al.* [16] and Boneh-Boyan (BB) signature of [3] are employed in our construction. Besides, interactive zero-knowledge proofs [12] are used. The adversarial model considered in this paper is static corruption model in which an adversary corrupts a party before the execution of the protocol. Corrupted parties do not follow the protocol specifications, remain corrupt throughout and are controlled by the adversary. Honest parties follow the protocol instructions. To control the malicious behavior of the parties, BB [3] signature is used. The sender computes the BB signature on the index of each message in initialization phase. Later in transfer phase, BB signature helps one to check whether the receiver has queried the valid ciphertext. As access polices are hidden in our construction, therefore, we first convert Ibraimi *et al.*'s protocol in to policy hiding CP-ABE which is then used to encrypt each message $m_i \in \mathsf{DB} = \{(m_i, \mathsf{AP}_i)\}_{1\leq i\leq N}$ under access policy AP_i to generate ciphertext database $\mathsf{cDB} = \{\Phi_i\}_{1\leq i\leq N}$. The policy hiding CP-ABE hides the access policies associated with each message and allows only authorized receivers to correctly decrypt the ciphertext. Authorized receivers are those whose attribute sets satisfy the access policies attached with the messages.

The security analysis is done in *full-simulation* model following [5,7] in which the sender's security and the receiver's security follow *real/ideal world* paradigm. In real world, parties interact with each other and follow the protocol instructions. While in ideal world, parties do not interact with each other. They interact via an incorruptible third party which is programmed to do all the computation work. Parties give their inputs to the trusted party and get back their respective outputs. A distinguisher distinguishes the output of both the worlds. In this model, hidden secret in zero-knowledge proofs are extracted by the simulator following adversarial rewinding that allows the simulator to rewind the adversary's state to previous computation and start the computation from there. The proposed AOT-HAP is secure assuming the hardness of Decision Bilinear Diffie-Hellman (DBDH), q-Strong Diffie-Hellman (SDH) [3] and q-Power Decisional Diffie-Hellman (PDDH) [9] problems. Our proposed AOT-HAP guarantees following security requirements.

1. The sender does not learn who queries a message and which message is being queried.
2. The receiver learns only one message in each query.
3. The receiver learns nothing about the access policies associated with the messages.
4. The receiver learns only those messages for which its attribute set satisfy the access policy attached with the message.

The proposed AOT-HAP realizes conjunction ($\wedge$) as well as disjunction ($\vee$) of attribution in comparison to [5,7] which cover only conjunction of attributes. More interestingly, the proposed AOT-HAP outperforms significantly in terms of both computation and communication overheads as compared to [5,7].

2 Preliminaries

Notations: Throughout, we use ρ as the security parameter, $x \xleftarrow{\$} A$ means sample an element x uniformly at random from the set A, $y \leftarrow B$ indicates y is the output of algorithm B, $X \overset{c}{\approx} Y$ denotes distribution X is computationally indistinguishable from distribution Y, $\Omega = \{a_1, a_2, \ldots, a_m\}$ denotes universe of attributes, $\mathcal{R} = \{1, 2, \ldots, n\}$ is universe of receivers, ID_R is an identity of a receiver $R \in \mathcal{R}$ and $\mathbb{N}$ denotes the set of natural numbers. A function $f(t)$ is *negligible* if $f = o(t^{-c})$ for every fixed positive constant c.

Definition 1 (Access Policy). *Let $\Omega = \{a_1, a_2, \ldots, a_m\}$ be the universe of attributes and $\mathcal{P}(\Omega)$ be the collection of all subsets of Ω. An access policy (structure) is a collection $\mathbb{A}$ of non-empty subsets of Ω, i.e., $\mathbb{A} \subseteq \mathcal{P}(\Omega)\backslash\emptyset$. The sets in $\mathbb{A}$ are called the authorized sets, and the sets not in $\mathbb{A}$ are called the unauthorized sets.*

2.1 Bilinear Pairing and Complexity Assumptions

Definition 2 (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 groups $\mathbb{G}_1$ and $\mathbb{G}_2$ respectively. Then the map $e : \mathbb{G}_1 \times \mathbb{G}_2 \rightarrow \mathbb{G}_T$ 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.

Definition 3 (q-SDH [3]). *The q-Strong Diffie-Hellman (SDH) assumption in $\mathbb{G}$ states that for all PPT algorithm $\mathcal{A}$, with running time in ρ, the advantage*

$$\mathsf{Adv}_{\mathbb{G}}^{q\text{-}SDH}(\mathcal{A}) = \mathsf{Pr}[\mathcal{A}(g, g^x, g^{x^2}, \ldots, g^{x^q}) = (c, g^{\frac{1}{x+c}})]$$

is negligible in ρ, where $g \xleftarrow{\$} \mathbb{G}, x \xleftarrow{\$} \mathbb{Z}_p, c \in \mathbb{Z}_p$.

Definition 4 (q-PDDH [9]). *The q-PDDH assumption in $(\mathbb{G}, \mathbb{G}_T)$ states that for all PPT algorithm $\mathcal{A}$, with running time in ρ, the advantage*

$$\mathsf{Adv}_{\mathbb{G},\mathbb{G}_T}^{q-PDDH}(\mathcal{A}) = \mathsf{Pr}[\mathcal{A}(g, g^x, g^{x^2}, \ldots, g^{x^q}, H, W)] - \mathsf{Pr}[\mathcal{A}(g, g^x, g^{x^2}, \ldots, g^{x^q}, H, V)]$$

is negligible in ρ, where $W = (H^x, H^{x^2}, \ldots, H^{x^q})$, $V = (V_1, V_2, \ldots, V_q)$, $g \xleftarrow{\$} \mathbb{G}, H, V_1, V_2, \ldots, V_q \xleftarrow{\$} \mathbb{G}_T, x \in \mathbb{Z}_p$.

Definition 5 (DBDH [16]). *The Decision Bilinear Diffie-Hellman (DBDH) assumption in $(\mathbb{G}, \mathbb{G}_T)$ states that for all PPT algorithm $\mathcal{A}$, with running time in ρ, the advantage*

$$\mathsf{Adv}_{\mathbb{G},\mathbb{G}_T}^{DBDH}(\mathcal{A}) = \mathsf{Pr}[\mathcal{A}(g, g^a, g^b, g^c, e(g, g)^{abc})] - \mathsf{Pr}[\mathcal{A}(g, g^a, g^b, g^c, Z)]$$

is negligible in ρ, where $g \xleftarrow{\$} \mathbb{G}, Z \xleftarrow{\$} \mathbb{G}_T, a, b, c \in \mathbb{Z}_p$.

2.2 Zero-knowledge Proof of Knowledge

Interactive zero-knowledge proof of knowledge introduced by [2] is a two-party interactive protocol between a prover and a verifier. We use the notation of [10] for the various zero-knowledge proofs of knowledge of discrete logarithms and proofs of validity of statements about discrete logarithms. For instance,

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

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 three properties.

- **Completeness:** If the statement is true, the honest verifier will accept the proof with high probability.
- **Soundness:** If the statement is false, the honest verifier will accept the proof with negligible probability.
- **Zero-knowledge:** If the statement is true, the verifier does not learn anything other than the fact.

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 Eq. 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 $\mathbb{Z}_p$.

2.3 Formal Model and Security Notions

The adaptive oblivious transfer with hidden access policy ($\mathsf{AOT\text{-}HAP}$) 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 with an identity ID_R has an attribute set w_{ID_R}. The issuer generates public/secret key pair to provide attribute secret keys corresponding to attribute sets of the receivers. The $\mathsf{AOT\text{-}HAP}$ completes in one *initialization phase*, one *issue phase* and k *transfer phases.* In initialization phase, S encrypts each message m_i of DB associated with AP_i in order to generate ciphertext database $\mathsf{cDB} = (\varPhi_1, \varPhi_2, \ldots, \varPhi_N)$. The sender S embeds access policies in cDB. In issue phase, each receiver R with an attribute set w_{ID_R} interacts with the issuer to get attribute secret key $\mathsf{ASK}_{w_{\mathsf{ID}_R}}$. In transfer phase, R interacts with S and recovers k messages of its choice sequentially. In each transfer phase, R has input $\sigma_j \in [1, N]$ and recovers m_{σ_j} after interacting with S, where $j = 1, 2, \ldots, k$.

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 algorithm encrypts the database DB in order to generate ciphertext database $\mathsf{cDB} = (\varPhi_1, \varPhi_2, \ldots, \varPhi_N)$, where access policy AP_i is not embedded explicitly in the corresponding ciphertext $\varPhi_i, i = 1, 2, \ldots, N$. 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 $\mathsf{ID}_R \in \mathcal{R}$ and attribute set $w_{\mathsf{ID}_R} \subseteq \varOmega$ 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 $\varPhi_\sigma$ 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 $\perp$.

Note 1. 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 $\varPhi$ 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 $\varPhi$ 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 [7]. 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 $\varPi$. 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 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 $\mathsf{DBSetup}$ 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 Issue protocol with the issuer on input ID_R and attribute set w_{ID_R}. After completion of Issue protocol, R returns (issue, 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 (encDB, 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 $\mathsf{DBInitialization}$ 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 $\mathsf{Transfer}$ 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}), (encDB, 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 $\mathsf{ID}_R \in \mathcal{R}$, 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 $\mathsf{ASK}_{w_{\mathsf{ID}_R}}$ corresponding to an attribute set w_{ID_R} of a receiver R with identity 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 (encDB, 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

$$\mathsf{REAL}_{\Pi,\mathcal{A},\mathcal{Z}} \overset{c}{\approx} \mathsf{IDEAL}_{\mathcal{F},\mathcal{A}',\mathcal{Z}}.$$

2.4 The BB Signature [3]

The Boneh and Boyen (BB) signature is used in our construction to sign the index of each message, and it consists of $\mathsf{BBSetup}$, $\mathsf{BBKeyGen}$, BBSign and $\mathsf{BBVerify}$ algorithms.

- $\mathsf{BBSetup}(1^\rho)$: Generate $\mathsf{params} = (p, \mathbb{G}, \mathbb{G}_T, e, g) \leftarrow \mathsf{BilinearSetup}(1^\rho)$, where $\mathsf{BilinearSetup}$ is an algorithm which on input security parameter ρ generates $\mathsf{params} = (p, \mathbb{G}, \mathbb{G}_T, e, g)$, where $e : \mathbb{G} \times \mathbb{G} \rightarrow \mathbb{G}_T$ 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.
- $\mathsf{BBKeyGen}(\mathsf{params})$: Pick $x \xleftarrow{\$} \mathbb{Z}_p^*$ and set $y = g^x$. The public key is $\mathsf{pk} = (g, y)$ and secret key is $\mathsf{sk} = x$.
- $\mathsf{BBSign}(\mathsf{sk}, \theta)$: The signature on message $\theta \in \mathbb{Z}_p$ is $\sigma = g^{\frac{1}{x+\theta}}$.
- $\mathsf{BBVerify}(\mathsf{pk}, \sigma, \theta)$: It outputs valid if $e(\sigma, y \cdot g^\theta) = e(g, g)$, otherwise, $\mathsf{invalid}$.

The signature scheme is existentially unforgeable under weak chosen-message attack (WEU) assuming the q-SDH is hard.

3 Concrete Construction

A high level description of our adaptive oblivious transfer protocol with hidden access policy ($\mathsf{AOT\text{-}HAP}$) 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 [16] 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 an 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 given in Sect. 2.4 which on input security parameter ρ generates $\mathsf{params} = (p, \mathbb{G}, \mathbb{G}_T, e, g)$.

- Isetup: The issuer on input ρ generates $\mathsf{params} \leftarrow \mathsf{BilinearSetup}(1^\rho)$, where $\mathsf{params} = (p, \mathbb{G}, \mathbb{G}_T, e, g)$. It picks $\alpha, t_1, t_2, \ldots, t_m \overset{\$}{\leftarrow} \mathbb{Z}_p^*$ and computes
$$Y = e(g, g)^\alpha, \quad T_j = g^{t_j}, j = 1, 2, \ldots, m.$$
The public/secret key pair is
$$\mathsf{PK}_I = (\mathsf{params}, Y, T_1, T_2, \ldots, T_m), \quad \mathsf{SK}_I = (\alpha, t_1, t_2, \ldots, t_m).$$
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 $x, \beta, \gamma \overset{\$}{\leftarrow} \mathbb{Z}_p^*$, $h \overset{\$}{\leftarrow} \mathbb{G}$ and sets
$$y = g^x, H = e(g, h), Z = e(g, g)^\beta, P = e(g, g)^\gamma.$$
The public/secret key pair is
$$\mathsf{pk}_{\mathsf{DB}} = (H, Z, P, y), \quad \mathsf{sk}_{\mathsf{DB}} = (h, x, \beta, \gamma).$$
The sender S publishes $\mathsf{pk}_{\mathsf{DB}}$ to all parties and keeps $\mathsf{sk}_{\mathsf{DB}}$ 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 $\mathsf{pk}_{\mathsf{DB}}$ checks the correctness of $\mathsf{pk}_{\mathsf{DB}}$ by verifying the POK. If it fails, R aborts the execution. Otherwise, R accepts $\mathsf{pk}_{\mathsf{DB}}$.
- DBInitialization: The sender S on input $\mathsf{PK}_I, \mathsf{params}, \mathsf{pk}_{\mathsf{DB}}, \mathsf{sk}_{\mathsf{DB}}$ and DB computes ciphertext database $\mathsf{cDB} = (\Phi_1, \Phi_2, \ldots, \Phi_N)$, $\mathsf{DB} = \{(m_i, \mathsf{AP}_i)\}_{1 \le i \le N}$, $m_i \in \mathbb{G}_T, i = 1, 2, \ldots, N$. Each message m_i is associated with an access policy AP_i. The ciphertext Φ_i for each message m_i is generated by S as follows. For $i = 1, 2, \ldots, N$, do

1. Parse params to extract g and $\mathsf{sk}_{\mathsf{DB}}$ 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 $s_i - \sum_{i=1}^{\ell-1} r_{i,j}$ as shown in Fig. 1. 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 as shown in Fig. 2.
      (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 $K_{i,j}^{(2)} = T_j^{\gamma s_{i,j}} \cdot g^{z_j}, \quad s_{i,j}, z_j \xleftarrow{\$} \mathbb{Z}_p^*$.
   (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.* [16] 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

$$e(A_i, y \cdot g^i) = e(g, g), \quad \forall\ i = 1, 2, \ldots, N,$$

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 $\mathsf{ID}_R \in \mathcal{R}$. The issuer

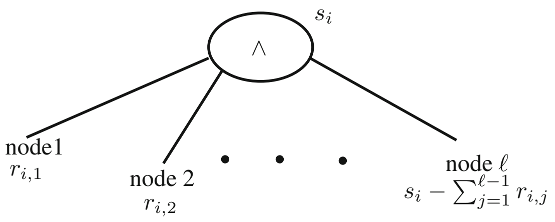

Fig. 1. Internal Node is "AND"

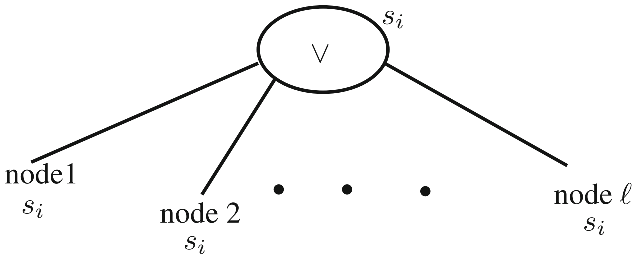

Fig. 2. Internal Node is "OR"

picks $r_{\mathsf{ID}_R} \xleftarrow{\$} \mathbb{Z}_p^*$ and sets

$$d_0 = g^{\alpha - r_{\mathsf{ID}_R}},\ d_l = g^{r_{\mathsf{ID}_R} \cdot t_l^{-l}}\ \forall\ a_l \in w_{\mathsf{ID}_R}.$$

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

$$\mathsf{POK}\{(\alpha, t_1, t_2, \ldots, t_m) | Y = e(g,g)^\alpha \wedge T_1 = g^{t_1} \wedge T_2 = g^{t_2} \wedge \ldots \wedge T_m = g^{t_m}\}$$

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 Fig. 3. 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

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

Sender(DB)		Receiver(ID_R)
		$\mathsf{SK}_{w_{\mathsf{ID}_R}} = (d_0, d_l\ \forall a_l \in w_{\mathsf{ID}_R})$
		$\sigma_j, j = 1, 2, \ldots, k$
		$\phi_{\sigma_j} = (A_{\sigma_j}, D_{\sigma_j}, F_{\sigma_j}, \pi_{\sigma_j})$
		$D_{\sigma_j} = (K^{(0)}_{\sigma_j}, K^{(1)}_{\sigma_j}, K^{(2)}_{\sigma_j,l}),$
		$l = 1, 2, \ldots, m$
		$F_{\sigma_j} = (Q_{\sigma_j}, L_{\sigma_j,1}, L_{\sigma_j,2}, \ldots, L_{\sigma_j,m})$
		$v_{\sigma_j} \xleftarrow{\$} \mathbb{Z}_p^*$
		$I_{\sigma_j} = e(K^{(1)}_{\sigma_j}, d_0)$
		$J_{\sigma_j} = \prod_{a_l \in w_{\mathsf{ID}_R}} e\left(K^{(2)}_{\sigma_j,l}, d_l\right)$
		$V_{\sigma_j} = A^{v_{\sigma_j}}_{\sigma_j}, X_{\sigma_j} = I^{v_{\sigma_j}}_{\sigma_j}, U_{\sigma_j} = J^{v_{\sigma_j}}_{\sigma_j}$
	$\xleftarrow[\mathsf{POK}\{(\sigma_j, v_{\sigma_j}) \mid e(V_{\sigma_j}, y) = e(V_{\sigma_j}, g)^{-\sigma_j} e(g,g)^{v_{\sigma_j}}\}]{V_{\sigma_j}, X_{\sigma_j}, U_{\sigma_j}}$	
$W_{\sigma_j} = e(V_{\sigma_j}, h) \cdot X^{\frac{1}{\beta}}_{\sigma_j} U^{\frac{1}{\gamma}}_{\sigma_j}$		
	$\xrightarrow[\mathsf{POK}\{(\beta, h, \gamma) \mid W_{\sigma_j} = e(V_{\sigma_j}, h) \cdot X^{\frac{1}{\beta}}_{\sigma_j} U^{\frac{1}{\gamma}}_{\sigma_j} \wedge Z = e(g,g)^{\beta} \wedge H = e(g,h) \wedge P = e(g,g)^{\gamma}\}]{W_{\sigma_j}}$	
		$\dfrac{K^{(1)}_{\sigma_j}}{W^{\frac{1}{v_{\sigma_j}}}_{\sigma_j}} = m_{\sigma_j}$

Fig. 3. 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

$$V_{\sigma_j} = A^{v_{\sigma_j}}_{\sigma_j}, X_{\sigma_j} = I^{v_{\sigma_j}}_{\sigma_j}, U_{\sigma_j} = J^{v_{\sigma_j}}_{\sigma_j}$$

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

$$W_{\sigma_j} = e(V_{\sigma_j}, h) \cdot X^{\frac{1}{\beta}}_{\sigma_j} U^{\frac{1}{\gamma}}_{\sigma_j}$$

and gives it to R together with the zero-knowledge proof of knowledge

$$\mathsf{POK}\{(\beta, h, \gamma) |\ W_{\sigma_j} = e(V_{\sigma_j}, h) \cdot X^{\frac{1}{\beta}}_{\sigma_j} U^{\frac{1}{\gamma}}_{\sigma_j} \wedge H = e(g,h) \wedge Z = e(g,g)^{\beta} \wedge P = e(g,g)^{\gamma}\}.$$

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

$$\frac{K^{(1)}_{\sigma_j}}{W^{\frac{1}{v_{\sigma_j}}}_{\sigma_j}} = m_{\sigma_j}. \qquad (2)$$

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

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

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 2. The CP-ABE scheme of Ibraimi *et al.* [16] 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.* [16] which is $(K_i^{(0)}, K_i^{(1)}, K_{i,j}^{(2)})$, where

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

$s_{i,j}$ are taken according to step 3(b) of algorithm $\mathsf{DBInitialization}$. 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 Security Analysis

Theorem 1. *The adaptive oblivious transfer with hidden access policy (*$\mathsf{AOT\text{-}HAP}$*) decribed in Sect. 2.3 securely implements the* $\mathsf{AOT\text{-}HAP}$ *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 $\mathsf{Game}\ i$ 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 $\mathsf{POK}\{(\alpha, t_1, t_2, \ldots, t_m) | Y = e(g,g)^{\alpha} \wedge T_1 = g^{t_1} \wedge T_2 = g^{t_2} \wedge \ldots \wedge T_m = g^{t_m}\}$ when the issue query is instructed by $\mathcal{Z}$. The difference between $\mathsf{Game}\ 1$ and $\mathsf{Game}\ 0$ is given by the knowledge error of POK which is negligible provided the underlying POK is sound. Therefore, there exists a negligible function $\epsilon_1(\rho)$ such that $|\mathsf{Pr}[\mathsf{Game}\ 1] - \mathsf{Pr}[\mathsf{Game}\ 0]| \leq \epsilon_1(\rho)$.

Game **2**: This game is the same as $\mathsf{Game}\ 1$ 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 between $\mathsf{Game}\ 2$ and $\mathsf{Game}\ 1$ is the knowledge error of POK which is negligible provided the underlying POK is sound. Therefore, there exists a negligible function $\epsilon_2(\rho)$ such that $|\mathsf{Pr}[\mathsf{Game}\ 2] - \mathsf{Pr}[\mathsf{Game}\ 1]| \leq \epsilon_2(\rho)$.

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 $\mathsf{Game}\ 3$ and $\mathsf{Game}\ 2$ is given by the knowledge error of POK which is negligible provided the underlying POK is sound. Therefore, there exists a negligible function $\epsilon_3(\rho)$ such that $|\mathsf{Pr}[\mathsf{Game}\ 3] - \mathsf{Pr}[\mathsf{Game}\ 2]| \leq \epsilon_3(\rho)$.

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 $\mathsf{Game}\ 4$ and $\mathsf{Game}\ 3$ 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 $\epsilon_4(\rho)$ such that $|\mathsf{Pr}[\mathsf{Game}\ 4] - \mathsf{Pr}[\mathsf{Game}\ 3]| \leq \epsilon_4(\rho)$.

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 $\mathsf{POK}\{(\alpha, t_1, t_2, \ldots, t_m) | Y = e(g,g)^\alpha \wedge T_1 = g^{t_1} \wedge T_2 = g^{t_2} \wedge \ldots \wedge T_m = g^{t_m}\}$ when the issue query is instructed by $\mathcal{Z}$. 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 $K_{i,l}^{(2)} = (L_{i,l})^{\gamma t_l}$, 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

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

Note that the Eq. 3 can be viewed as an instance of the knapsack problem as finding a solution of the Eq. 3 is essentially the same as finding solution of $\sum_{t \in \mathcal{I}_i} s_{i,t} x_t = s_i$, where $x_t \in \{0,1\}$, $\mathcal{I}_i = \{t \mid a_t \in \mathsf{atr}_i\}$ 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 Eq. 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 $\mathsf{IDEAL}_{\mathcal{F},\mathcal{Z},\mathcal{A}'} = \mathsf{Pr}[\mathsf{Game}\ 4] + \epsilon_{knap}$ and $\mathsf{IDEAL}_{\mathcal{F},\mathcal{Z},\mathcal{A}'} - \mathsf{REAL}_{\Pi,\mathcal{Z},\mathcal{A}} = |\mathsf{Pr}[\mathsf{Game}\ 4] - [\mathsf{Game}\ 0]| + \epsilon_{knap} \leq |\mathsf{Pr}[\mathsf{Game}\ 4] - [\mathsf{Game}\ 3]| + |\mathsf{Pr}[\mathsf{Game}\ 3] - [\mathsf{Game}\ 2]| + |\mathsf{Pr}[\mathsf{Game}\ 2] - [\mathsf{Game}\ 1]| + |\mathsf{Pr}[\mathsf{Game}\ 1] - [\mathsf{Game}\ 0]| + \epsilon_{knap} \leq \epsilon_4(\rho) + \epsilon_3(\rho) + \epsilon_2(\rho) + \epsilon_1(\rho) + \epsilon_{knap} = \nu(\rho)$, 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, $\mathsf{Pr}[\mathsf{Game}\ 0] = 0$. Therefore, $\mathsf{REAL}_{\Pi,\mathcal{Z},\mathcal{A}} = \mathsf{Pr}[\mathsf{Game}\ 0]$.

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 $\epsilon_1(\rho)$ such that $|\mathsf{Pr}[\mathsf{Game}\ 1] - \mathsf{Pr}[\mathsf{Game}\ 0]| \le \epsilon_1(\rho)$.

Game **2**: In this game, the simulator $\mathcal{S}_2$ computes $\widehat{A_{\sigma_j}} = V_{\sigma_j}^{\frac{1}{v_{\sigma_j}}}$ and

$$\widehat{e(g, K_{\sigma_j}^{(1)})}^{\frac{1}{\beta}} = X_{\sigma_j}^{\frac{1}{\beta v_{\sigma_j}}} U_{\sigma_j}^{\frac{1}{\gamma v_{\sigma_j}}}$$

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 $\widehat{e(g, K_{\sigma_j}^{(1)})}^{\frac{1}{\beta}} = e(g, K_{\sigma_j}^{(1)})^{\frac{1}{\beta}}$ 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 [3].

Hence, there exists a negligible function $\epsilon_2(\rho)$ such that $|\mathsf{Pr}[\mathsf{Game}\ 2] - \mathsf{Pr}[\mathsf{Game}\ 1]| \le \epsilon_2(\rho)$.

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 $\epsilon_3(\rho)$ such that $|\mathsf{Pr}[\mathsf{Game}\ 3] - \mathsf{Pr}[\mathsf{Game}\ 2]| \le \epsilon_3(\rho)$.

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 $\epsilon_4(\rho)$ such that $|\mathsf{Pr}[\mathsf{Game\ 4}]\text{-}\mathsf{Pr}[\mathsf{Game\ 3}]| \leq \epsilon_4(\rho)$.

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 $\mathbb{G}_T, i = 1, 2, \ldots, N$.

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 $\mathsf{IDEAL}_{\mathcal{F},\mathcal{Z},\mathcal{A}'} = \mathsf{Pr}[\mathsf{Game\ 4}]$ and $\mathsf{IDEAL}_{\mathcal{F},\mathcal{Z},\mathcal{A}'} - \mathsf{REAL}_{\Pi,\mathcal{Z},\mathcal{A}} = |\mathsf{Pr}[\mathsf{Game\ 4}] - [\mathsf{Game\ 0}]| \leq |\mathsf{Pr}[\mathsf{Game\ 4}] - [\mathsf{Game\ 3}]| + |\mathsf{Pr}[\mathsf{Game\ 3}] - [\mathsf{Game\ 2}]||\mathsf{Pr}[\mathsf{Game\ 2}] - [\mathsf{Game\ 1}]| + |\mathsf{Pr}[\mathsf{Game\ 1}] - [\mathsf{Game\ 0}]| \leq \epsilon_4(\rho) + \epsilon_3(\rho) + \epsilon_2(\rho) + \epsilon_1(\rho) = \nu(\rho)$, 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.

5 Comparison with AOT-HAP in [5,7]

In this section, we compare the proposed scheme with the AOT-HAP in [5,7] which are the only two AOT-HAP to the best of our knowledge. The proposal of [7] employed Camenisch *et al.'s* [9] oblivious transfer, batch Boneh-Boyan (BB) [3] signature and Camenisch-Lysyanskaya (CL) signature [8]. On the contrary, the AOT-HAP of Camenisch *et al.* [5] relies on interactive zero-knowledge proofs [12], Groth-Sahai non-interactive proofs [15], the privacy friendly signature [1] and ciphertext-policy attribute-based encryption (CP-ABE) [20]. We point that in [7], 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 [5] 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 [5] has n_i values whereas in [7] each category c_i has only two values. The schemes in [5,7] covers only conjunction of attributes. In contrast to [5,7], our scheme employs modified policy hiding Ibraimi *et al.*'s [16] CP-ABE and BB [3] 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 [5,7] as illustrated in Tables 1, 2 and 3, 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 [7], $m = 2l$ and in [5] $m = n_1 + n_2 + \ldots + n_l$. Note that [5,7] used asymmetric bilinear pairing $e : \mathbb{G}_1 \times \mathbb{G}_2 \rightarrow \mathbb{G}_T$. 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 [5,7] 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 $\mathsf{pk_{DB}}$ in DBSetup algorithm while that of [5,7] 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 [5,7] is linear to l. Total number of exponentiations is less in our scheme as compared to [5,7]. Communication-wise our construction performs favorably over [5,7]. Table 3 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.

Table 1. 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
[5]	$(m + 4l + 16)\mathbb{G}_1 + (4l + 10)\mathbb{G}_2$	1	–	$4\mathbb{G}_2$	–	$8l + 26$
[7]	$(4l + 7)\mathbb{G}_1 + (8l + 9)\mathbb{G}_2$	–	1	–	$16l + 10$	$144l + 19$
Ours	$(3m + 2)\mathbb{G}$	3	1	$(2m + 1)\mathbb{G}$	2	1

Table 2. 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
[5]	$17\mathbb{G}_1 + 9\mathbb{G}_2$	45	41	$27\mathbb{G}_1 + 22\mathbb{G}_2$	38	$2l + 43$
[7]	$(12l + 104)\mathbb{G}_1$	$4l + 51$	$18l + 20$	$(16l + 99)\mathbb{G}_1 + (8l + 35)\mathbb{G}_2$	$3l + 36$	1
Ours	–	9	4	$1\mathbb{G}$	14	6

Table 3. Comparison in terms of communication.

AOT-HAP	Per ciphertext			Per Transfer protocol		
	$\mathbb{Z}_p$	$\mathbb{G}_1 + \mathbb{G}_2$	$\mathbb{G}_T$	$\mathbb{Z}_p$	$\mathbb{G}_1 + \mathbb{G}_2$	$\mathbb{G}_T$
1-7 [5]	1	$(m+2l+11)\mathbb{G}_1 + (2l + 4)\mathbb{G}_2$	1	19	$18\mathbb{G}_1 + 14\mathbb{G}_2$	21
[7]	–	$(4l + 2)\mathbb{G}_1 + (4l + 5)\mathbb{G}_2$	1	$(6l + 55)\mathbb{G}_1 + (4l + 40)\mathbb{G}_2$	$4l + 7$	17
Ours	$2m + 2$	$3m + 2$	3	8	2	8

The communication cost involved in transferring elements the from issuer to R and from R to the issuer is significantly low as compared to [5,7].

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 as compared to existing similar schemes.

References

1. Abe, M., Fuchsbauer, G., Groth, J., Haralambiev, K., Ohkubo, M.: Structure-preserving signatures and commitments to group elements. In: Rabin, T. (ed.) CRYPTO 2010. LNCS, vol. 6223, pp. 209–236. Springer, Heidelberg (2010)
2. Bellare, M., Goldreich, O.: On defining proofs of knowledge. In: Brickell, E.F. (ed.) CRYPTO 1992. LNCS, vol. 740, pp. 390–420. Springer, Heidelberg (1993)
3. Boneh, D., Boyen, X.: Short signatures without random oracles. In: Cachin, C., Camenisch, J.L. (eds.) EUROCRYPT 2004. LNCS, vol. 3027, pp. 56–73. Springer, Heidelberg (2004)
4. Brassard, G., Crépeau, C., Robert, J.M.: All-or-nothing disclosure of secrets. In: Odlyzko, A.M. (ed.) CRYPTO 1986. LNCS, vol. 263, pp. 234–238. Springer, Heidelberg (1987)
5. Camenisch, J., Dubovitskaya, M., Enderlein, R.R., Neven, G.: Oblivious transfer with hidden access control from attribute-based encryption. In: Visconti, I., De Prisco, R. (eds.) SCN 2012. LNCS, vol. 7485, pp. 559–579. Springer, Heidelberg (2012)
6. Camenisch, J., Dubovitskaya, M., Neven, G.: Oblivious transfer with access control. In: ACM 2009, pp. 131–140. ACM (2009)
7. Camenisch, J., Dubovitskaya, M., Neven, G., Zaverucha, G.M.: Oblivious transfer with hidden access control policies. In: Catalano, D., Fazio, N., Gennaro, R., Nicolosi, A. (eds.) PKC 2011. LNCS, vol. 6571, pp. 192–209. Springer, Heidelberg (2011)
8. Camenisch, J.L., Lysyanskaya, A.: Signature schemes and anonymous credentials from bilinear maps. In: Franklin, M. (ed.) CRYPTO 2004. LNCS, vol. 3152, pp. 56–72. Springer, Heidelberg (2004)
9. Camenisch, J.L., Neven, G., Shelat, A.: Simulatable adaptive oblivious transfer. In: Naor, M. (ed.) EUROCRYPT 2007. LNCS, vol. 4515, pp. 573–590. Springer, Heidelberg (2007)
10. Camenisch, J.L., Stadler, M.A.: Efficient group signature schemes for large groups. In: Kaliski Jr., B.S. (ed.) CRYPTO 1997. LNCS, vol. 1294, pp. 410–424. Springer, Heidelberg (1997)
11. 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)
12. Cramer, R., Damgård, I.B., MacKenzie, P.D.: Efficient zero-knowledge proofs of knowledge without intractability assumptions. In: Imai, H., Zheng, Y. (eds.) PKC 2000. LNCS, vol. 1751, pp. 354–373. Springer, Heidelberg (2000)
13. Green, M., Hohenberger, S.: Blind identity-based encryption and simulatable oblivious transfer. In: Kurosawa, K. (ed.) ASIACRYPT 2007. LNCS, vol. 4833, pp. 265–282. Springer, Heidelberg (2007)
14. Green, M., Hohenberger, S.: Universally composable adaptive oblivious transfer. In: Pieprzyk, J. (ed.) ASIACRYPT 2008. LNCS, vol. 5350, pp. 179–197. Springer, Heidelberg (2008)
15. Groth, J., Sahai, A.: Efficient non-interactive proof systems for bilinear groups. In: Smart, N.P. (ed.) EUROCRYPT 2008. LNCS, vol. 4965, pp. 415–432. Springer, Heidelberg (2008)

16. Ibraimi, L., Tang, Q., Hartel, P., Jonker, W.: Efficient and provable secure ciphertext-policy attribute-based encryption schemes. In: Bao, F., Li, H., Wang, G. (eds.) ISPEC 2009. LNCS, vol. 5451, pp. 1–12. Springer, Heidelberg (2009)
17. Naor, M., Pinkas, B.: Oblivious transfer and polynomial evaluation. In: TOC 1999, pp. 245–254. ACM (1999)
18. Naor, M., Pinkas, B.: Efficient oblivious transfer protocols. In: SODA 2001, pp. 448–457. Society for Industrial and Applied Mathematics (2001)
19. Naor, M., Pinkas, B.: Computationally secure oblivious transfer. J. Cryptology **18**(1), 1–35 (2005)
20. Nishide, T., Yoneyama, K., Ohta, K.: Attribute-based encryption with partially hidden encryptor-specified access structures. In: Bellovin, S.M., Gennaro, R., Keromytis, A.D., Yung, M. (eds.) ACNS 2008. LNCS, vol. 5037, pp. 111–129. Springer, Heidelberg (2008)
21. Peikert, C., Vaikuntanathan, V., Waters, B.: A framework for efficient and composable oblivious transfer. In: Wagner, D. (ed.) CRYPTO 2008. LNCS, vol. 5157, pp. 554–571. Springer, Heidelberg (2008)
22. Rabin, M.O.: How to exchange secrets by oblivious transfer. Technical report TR-81, Harvard Aiken Computation Laboratory (1981)
23. Zhang, Y., Au, M.H., Wong, D.S., Huang, Q., Mamoulis, N., Cheung, D.W., Yiu, S.-M.: Oblivious transfer with access control: realizing disjunction without duplication. In: Joye, M., Miyaji, A., Otsuka, A. (eds.) Pairing 2010. LNCS, vol. 6487, pp. 96–115. Springer, Heidelberg (2010)