I. Security Proof

The security of the proposed PIB-MKEM scheme is guaranteed by following lemmas.

Lemma 1. *If the BCDH assumption holds over the bilinear group $(e, p, g, \mathbb{G}, \mathbb{G}_T)$, then the proposed PIB-MKEM scheme is IND-MIS-CPA secure in the random oracle model.*

Proof. Throughout the proof, we will demonstrate that if there exists a PPT adversary $\mathcal{A}$ that can break the IND-MIS-CPA security of the proposed PIB-MKEM with a non-negligible advantage, then we can construct another PPT simulator C that can also break the BCDH assumption with a non-negligible advantage. Specifically, this is achieved by letting C simulates the security experiment played with $\mathcal{A}$ as follows.

Setup phase: Initially, C receives an instance (g, g^a, g^b, g^c) of the BCDH problem over bilinear groups $(e, p, g, \mathbb{G}, \mathbb{G}_T)$. Its goal is to compute $D = e(g, g)^{abc}$. Then, it generates a Bloom filter $(H, L) \leftarrow \mathsf{BFGen}(m, k)$, and lets $g_1 = g^a$. In addition, it respectively simulates three random oracles $G : \{0,1\}^* \to \mathbb{G}$, $G' : \{0,1\}^* \to \mathbb{G}$ and $\tilde{G} : \mathbb{G}_T \to \{0,1\}^\ell$ by maintaining three lists $\mathcal{L}_G$, $\mathcal{L}_{G'}$ and $\mathcal{L}_{\tilde{G}}$. Finally, it forwards the public parameter $\mathsf{pp} = \{e, p, g, g_1, \mathbb{G}, \mathbb{G}_T, H, G, G', \tilde{G}\}$ to $\mathcal{A}$, and implicitly assigns the master secret key as $\mathsf{msk} = \{a, b\}$.

Query phase: To answer queries issued by the adversary $\mathcal{A}$, the simulator C has to simulate responses from random oracles G, G' and $\tilde{G}$. Specifically, C responds as follows:

- $G(\mathsf{id_r}||i)$: If there has been a tuple $(\mathsf{id_r}, i, Q, x, \gamma) \in \mathcal{L}_G$, then C directly returns Q as the response. Otherwise, it picks a random bit $\gamma \in \{0,1\}$ such that $\Pr[\gamma = 0] = \sigma$, and chooses a random exponent $x \in \mathbb{Z}_p$. In the case of $\gamma = 0$, it lets $Q = g^x$, and adds the tuple $(\mathsf{id_r}, i, Q, x, 0)$ to $\mathcal{L}_G$. In the case of $\gamma = 1$, it computes $Q = (g^b)^x$, and adds the tuple $(\mathsf{id_r}, i, Q, x, 1)$ to $\mathcal{L}_G$. Finally, C returns Q to $\mathcal{A}$.
- $G'(\mathsf{id_s})$: If there has been a tuple $(\mathsf{id_s}, r, R) \in \mathcal{L}_{G'}$, then C directly returns R as the response. Otherwise, it randomly picks an integer $r \in \mathbb{Z}_p$, computes $R = g^r$, adds the tuple $(\mathsf{id_s}, r, R)$ to $\mathcal{L}_{G'}$, and returns R to $\mathcal{A}$.
- $\tilde{G}(w)$: If there has been a tuple $(w, \mathsf{W}) \in \mathcal{L}_{\tilde{G}}$, then C directly returns W as the response. Otherwise, it randomly selects a binary string $\mathsf{W} \in \{0,1\}^\ell$, adds the tuple (w, W) to $\mathcal{L}_{\tilde{G}}$, and returns W to $\mathcal{A}$.

By invoking the above random oracles, the simulator C can answer the adversary $\mathcal{A}$'s queries in the following way:

- $\mathcal{Q}_{\mathsf{RKeyGen}}(\mathsf{id_r})$: A decapsulation key for $\mathsf{id_r}$ is assigned as $\mathsf{dk} = \{\mathsf{dk}_{i,1}, \mathsf{dk}_{i,2}\}_{i \in [m]} = \{G(\mathsf{id_r}||i)^a, G(\mathsf{id_r}||i)^b\}_{i \in [m]}$, associated with a binary string L. To produce such a key, C first retrieves the tuple $(\mathsf{id_r}, i, Q, x, \gamma) \in \mathcal{L}_G$ for each $i \in [m]$[1]. If $\gamma = 0$ holds for all these tuples, then it lets $\mathsf{dk}_{i,1} = Q^a = (g^a)^x$ and $\mathsf{dk}_{i,2} = Q^b = (g^b)^x$ for $i \in [m]$. Otherwise, C aborts the simulation. Finally, C returns the resulted decapsulation key $\mathsf{dk} = \{\mathsf{dk}_{i,1}, \mathsf{dk}_{i,2}\}_{i \in [m]}$ to $\mathcal{A}$.
- $\mathcal{Q}_{\mathsf{SKeyGen}}(\mathsf{id_s})$: Recall that an encapsulation key for $\mathsf{id_s}$ is computed as $\mathsf{ek} = G'(\mathsf{id_s})^b$. To generate such a key, C retrieves the tuple $(\mathsf{id_s}, r, R) \in \mathcal{L}_{G'}$, assigns and returns $\mathsf{ek} = R^b = (g^b)^r$ to $\mathcal{A}$.
- $\mathcal{Q}_{\mathsf{Punc}}(\mathsf{id_r}, \mathsf{ct})$: Whenever $\mathcal{A}$ issues such a query, C punctures the corresponding decapsulation key dk to dk' as in the original puncture algorithm, and also updates the triple to $(\mathsf{id_r}, \mathsf{dk}', \mathcal{P} \cup \{\mathsf{ct}\})$.

Challenge phase: The adversary $\mathcal{A}$ chooses and submits four identities $(\mathsf{id_{s0}}, \mathsf{id_{s1}}, \mathsf{id_{r0}}, \mathsf{id_{r1}})$ to the challenger C. After that, C generates the challenge ciphertext by conducting the following steps:

1) Select a random integer $v \in \mathbb{Z}_p$, compute $V = g^v$, and implicitly assign $U = g^c$.
2) For each $j \in [k]$, compute $\delta_j = H_j(U \cdot V)$, and retrieve the corresponding tuples $(\mathsf{id_{r0}}, \delta_j, Q_0, x_0, \gamma_0) \in \mathcal{L}_G$ and $(\mathsf{id_{r1}}, \delta_j, Q_1, x_1, \gamma_1) \in \mathcal{L}_G$. If either $\gamma_0 \neq 1$ or $\gamma_1 \neq 1$, abort the simulation. Otherwise, for $\theta \in \{0,1\}$, it holds that $G(\mathsf{id}_{\mathsf{r}\theta}||\delta_j) = (g^b)^{x_\theta}$.
3) Retrieve $(\mathsf{id_{s0}}, r_0, R_0) \in \mathcal{L}_{G'}$ and $(\mathsf{id_{s1}}, r_1, R_1) \in \mathcal{L}_{G'}$, and randomly pick two symmetric keys $\mathsf{K}_0, \mathsf{K}_1 \in \{0,1\}^\ell$. Then, choose a random bit $\beta \in \{0,1\}$ and random binary string $\mathsf{W}_j \in \{0,1\}^\ell$, and compute
$$r_j^* = e\big(G(\mathsf{id}_{\mathsf{r}\beta}||\delta_j), V \cdot (g^b)^{r_\beta}\big),\ c_j^* = \mathsf{K}_\beta \oplus \mathsf{W}_j \oplus \tilde{G}(r_j^*).$$
4) Return $(\mathsf{K}_\beta, \mathsf{ct}_\beta)$, where $\mathsf{ct}_\beta = \{U, V, \{c_j^*\}_{j \in [k]}\}$.

In particular, according to the decapsulation procedure, note that the decapsulation for ct^* is as follows:

$$\begin{aligned}\mathsf{K}_\beta &= c_j^* \oplus \tilde{G}\big(e(G(\mathsf{id}_{\mathsf{r}\beta}||\delta_j), \mathsf{ek}_\beta)\big) \oplus \tilde{G}\big(e(G(\mathsf{id}_{\mathsf{r}\beta}||\delta_j), g_1^c)\big)\\ &= c_j^* \oplus \tilde{G}\big(e(G(\mathsf{id}_{\mathsf{r}\beta}||\delta_j), V \cdot (g^b)^{r_\beta})\big) \oplus \tilde{G}(D^{x_\beta}),\end{aligned}$$

where $D = e(g, g)^{abc}$.

Guess phase: At this moment, the adversary $\mathcal{A}$ guesses and returns a bit $\beta' \in \{0,1\}$ to the simulator C. Then, C randomly retrieves a tuple $(\mathsf{id}_{\mathsf{r}\beta'}, j^*, Q_{\beta'}, x_{\beta'}, 1) \in \mathcal{L}_G$ for some integer $j^* \in [k]$, randomly selects a tuple $(w^*, \mathsf{W}^*) \in \mathcal{L}_{\tilde{G}}$, and outputs $(w^*)^{1/x_{\beta'}}$ as the final solution of the received instance of the BCDH problem.

Probability Analysis. Throughout the simulation, observe that the simulator C's responses to random oracles G, G' and $\tilde{G}$ are as in the real experiment, since each response is randomly and uniformly sampled from corresponding space. In addition, given implicitly assigned master secret key $\mathsf{msk} = \{a, b\}$, all responses to encapsulation and decapsulation key queries have correct distributions. Consequently, if C does not abort the simulation, then it perfectly simulates the security experiment in the view of the adversary $\mathcal{A}$. Below we bound the probability that C does not abort the simulation, and denote this event by E_0. Then, we capture the advantage of C solving the instance of the BCDH problem.

Assume that $\mathcal{A}$ makes a total of q_{dk} queries to $\mathcal{Q}_{\mathsf{RKeyGen}}(\cdot)$, then the probability that $\mathcal{A}$ does not abort in the query phases is $\sigma^{q_{dk}}$. Similarly, $\mathcal{A}$ does not abort in the challenge phase with probability $(1 - \sigma)^2$. Consequently, according to the analysis in [1] and [2], we have that

$$\Pr[\mathsf{E}_0] = \sigma^{q_{dk}} \cdot (1 - \sigma)^2,$$

[1]If there dose not exist such a tuple in the list $\mathcal{L}_G$, the simulator C accesses the random oracle $G(\cdot)$ to generate one. Subsequent similar situations are all handled in this way.

which is maximized at $\sigma = q_{dk}/(q_{dk}+2)$. If we employ this value as the probability of sampling $\gamma = 0$ from $\{0,1\}$ when responding the random oracle $G(\cdot)$, then we further have that

$$\Pr[\mathsf{E}_0] \geq \frac{4}{e^2(q_{dk}+2)^2},$$

where $e \approx 2.7$ is the base of the natural logarithm.

Conditioned on the occurrence of the event E_0, throughout the simulation, if $\mathcal{A}$ never issues a query to the random oracle $\tilde{G}(\cdot)$ with the input $e(G(\mathsf{id}_{\mathsf{r}\beta'}||\delta_j), g_1^c)$ for any $j \in [k]$, then it obtains no information about the symmetric key contained in the challenge ciphertext ct^*. Therefore, the probability of its guess being correct (i.e., $\beta' = \beta$) is $1/2$, and its advantage $\mathsf{Adv}^{\text{IND-MIS-CPA}}_{\mathcal{A},\text{PIB-MKEM}}(\lambda, m, k) = 0$. In this case, the lemma holds trivially.

On the other hand, if $\mathcal{A}$ has issued a query to the random oracle $\tilde{G}(\cdot)$ with the input $e(G(\mathsf{id}_{\mathsf{r}\beta'}||\delta_{j^*}), g_1^c)$ for some integer $j^* \in [k]$, then C correctly guesses $w^* = e(G(\mathsf{id}_{\mathsf{r}\beta'}||\delta_{j^*}), g_1^c)$ with probability $1/q_{\tilde{G}}$, where $q_{\tilde{G}}$ is the total number of queries issued to the random oracle $\tilde{G}(\cdot)$. We denote the event of its correct guess on j^* and w^* by E_1. In this case, C correctly resolves the instance of the BCDH problem as follows:

$$\begin{aligned} D &= (w^*)^{1/x_{\beta'}} = e(G(\mathsf{id}_{\mathsf{r}\beta'}||\delta_{j^*}), g_1^c)^{1/x_{\beta'}} \\ &= e\big((g^b)^{x_{\beta'}}, (g^a)^c\big)^{1/x_{\beta'}} = e(g,g)^{abc}. \end{aligned}$$

Furthermore, according to the analysis in [1], the probability that the event E_1 happens is bounded as follows:

$$\Pr[\mathsf{E}_1] \geq \frac{2 \cdot \mathsf{Adv}^{\text{IND-MIS-CPA}}_{\mathcal{A},\text{PIB-MKEM}}(\lambda, m, k)}{k \cdot q_{\tilde{G}}}.$$

Therefore, the advantage of C solving the instance of the BCDH problem is captured as follows:

$$\begin{aligned} \mathsf{Adv}^{\text{BCDH}}_{C}(\lambda) &= \Pr\left[C(g, g^a, g^b, g^c) = e(g,g)^{abc}\right] \\ &= \Pr[\mathsf{E}_0 \wedge \mathsf{E}_1] \\ &\geq \frac{8 \cdot \mathsf{Adv}^{\text{IND-MIS-CPA}}_{\mathcal{A},\text{PIB-MKEM}}(\lambda, m, k)}{e^2 \cdot k \cdot q_{\tilde{G}} \cdot (2+q_{dk})^2}. \end{aligned}$$

This completes the proof. □

Lemma 2. *If the BCDH assumption holds over the bilinear group $(e, p, g, \mathbb{G}, \mathbb{G}_T)$, then the proposed PIB-MKEM scheme is AUTH secure in the random oracle model.*

Proof. Similarly, we will show that if there exists a PPT adversary $\mathcal{A}$ that can break the AUTH security of our PIB-MKEM with a non-negligible advantage, then we can construct another PPT simulator C that can break the BCDH assumption with a non-negligible advantage. Specifically, C simulates the AUTH security experiment as follows.

Setup phase: Initially, C receives an instance (g, g^a, g^b, g^c) of the BCDH problem over bilinear groups $(e, p, g, \mathbb{G}, \mathbb{G}_T)$, and tries to compute $D = e(g,g)^{abc}$. To establish the system, C first produces a Bloom filter $(H, L) \leftarrow \mathsf{BFGen}(m, k)$, and lets $g_1 = g^a$. Then, it respectively simulates three random oracles $G : \{0,1\}^* \to \mathbb{G}$, $G' : \{0,1\}^* \to \mathbb{G}$ and $\tilde{G} : \mathbb{G}_T \to \{0,1\}^\ell$ by maintaining three lists $\mathcal{L}_G$, $\mathcal{L}_{G'}$ and $\mathcal{L}_{\tilde{G}}$. Finally, it forwards the public parameter $\mathsf{pp} = \{e, p, g, g_1, \mathbb{G}, \mathbb{G}_T, H, G, G', \tilde{G}\}$ to $\mathcal{A}$, and assigns the master secret key as $\mathsf{msk} = \{a, b\}$, which is unknown for C.

Query phase: The simulator C responds random oracles G, G' and $\tilde{G}$ as follows:

- $G(\mathsf{id}_\mathsf{r}||i)$: If there has been a tuple $(\mathsf{id}_\mathsf{r}, i, Q, x, \gamma) \in \mathcal{L}_G$, then C directly returns Q as the response. Otherwise, it picks a random bit $\gamma \in \{0,1\}$ such that $\Pr[\gamma = 0] = \sigma$, and chooses a random exponent $x \in \mathbb{Z}_p$. In the case of $\gamma = 0$, it lets $Q = g^x$, and adds the tuple $(\mathsf{id}_\mathsf{r}, i, Q, x, 0)$ to $\mathcal{L}_G$. In the case of $\gamma = 1$, it computes $Q = (g^c)^x$, and adds the tuple $(\mathsf{id}_\mathsf{r}, i, Q, x, 1)$ to $\mathcal{L}_G$. Finally, C returns Q to $\mathcal{A}$.
- $G'(\mathsf{id}_\mathsf{s})$: If there has been a tuple $(\mathsf{id}_\mathsf{s}, r, R, \zeta) \in \mathcal{L}_{G'}$, then C directly returns R as the response. Otherwise, it picks a random bit $\zeta \in \{0,1\}$ such that $\Pr[\zeta = 0] = \sigma$, and picks a random integer $r \in \mathbb{Z}_p$. In the case of $\zeta = 0$, it computes $R = g^r$, and adds the tuple $(\mathsf{id}_\mathsf{s}, r, R, 0)$ to $\mathcal{L}_{G'}$. In the case of $\zeta = 1$, it computes $R = (g^a)^r$, and adds the tuple $(\mathsf{id}_\mathsf{s}, r, R, 1)$ to $\mathcal{L}_{G'}$. Finally, C returns R to $\mathcal{A}$.
- $\tilde{G}(w)$: If there has already been a tuple $(w, \mathsf{W}) \in \mathcal{L}_{\tilde{G}}$, then C directly returns W as the response. Otherwise, it randomly chooses a binary string $\mathsf{W} \in \{0,1\}^\ell$, adds the tuple (w, W) to $\mathcal{L}_{\tilde{G}}$, and returns W to $\mathcal{A}$.

Based on the above random oracles, the simulator C answers the adversary $\mathcal{A}$'s key queries as follows:

- $Q_{\mathsf{RKeyGen}}(\mathsf{id}_\mathsf{r})$: For each index $i \in [m]$, C first retrieves the tuple $(\mathsf{id}_\mathsf{r}, i, Q, x, \gamma) \in \mathcal{L}_G$. If $\gamma = 0$ holds for all these tuples, then it directly computes $\mathsf{dk}_{i,1} = Q^a = (g^a)^x$ and $\mathsf{dk}_{i,2} = Q^b = (g^b)^x$ for $i \in [m]$. Otherwise, C aborts the simulation. Finally, C returns the resulted decryption key $\mathsf{dk} = \{\mathsf{dk}_{i,1}, \mathsf{dk}_{i,2}\}_{i\in[m]}$ to $\mathcal{A}$.
- $Q_{\mathsf{SKeyGen}}(\mathsf{id}_\mathsf{s})$: C retrieves the tuple $(\mathsf{id}_\mathsf{s}, r, R, \zeta) \in \mathcal{L}_{G'}$. If $\zeta = 1$ then C aborts the simulation. Otherwise, it assigns and returns $\mathsf{ek} = R^b = (g^b)^r$ to $\mathcal{A}$.
- $Q_{\mathsf{Punc}}(\mathsf{id}_\mathsf{r}, \mathsf{ct})$: Whenever $\mathcal{A}$ issues such a query, C punctures the corresponding decryption key dk to dk' as in the original puncture algorithm, and also updates the triple to $(\mathsf{id}_\mathsf{r}, \mathsf{dk}', \mathcal{P} \cup \{\mathsf{ct}\})$.

Forgery phase: The adversary $\mathcal{A}$ generates a forged symmetric key and ciphertext pair $(\mathsf{K}^*, \mathsf{ct}^*)$ under a sender's identity id_s^* and a receiver's identity id_r^*, and returns it to the simulator C. After that, C tries to compute $D = e(g,g)^{abc}$ as follows:

1) Parse the ciphertext ct^* as $\{U, V, \{c_j^*\}_{j\in[k]}\}$, randomly select an index $j^* \in [k]$, and compute $\delta_{j^*} = H_{j^*}(U \cdot V)$.
2) Respectively retrieve the tuples $(\mathsf{id}_\mathsf{s}^*, r, R, \zeta) \in \mathcal{L}_{G'}$ and $(\mathsf{id}_\mathsf{r}^*, \delta_{j^*}, Q, x, \gamma) \in \mathcal{L}_G$. If either $\gamma \neq 1$ or $\zeta \neq 1$, then C aborts the simulation. Otherwise, we have that the δ_{j^*}-th decapsulation key component for id_r^* is implicitly computed as $\mathsf{dk}_{\delta_{j^*},2} = G(\mathsf{id}_\mathsf{r}^*||\delta_{j^*})^b = (g^{bc})^x$ and $G'(\mathsf{id}_\mathsf{s}^*) = (g^a)^r$. This implies that

$$\begin{aligned} &\tilde{G}\big(e(\mathsf{dk}_{\delta_{j^*},2}, G'(\mathsf{id}_\mathsf{s}^*)) \cdot e(G(\mathsf{id}_\mathsf{r}^*||\delta_{j^*}), V)\big) \\ &= \tilde{G}\big(D^{x\cdot r} \cdot e(g^{c\cdot x}, V)\big), \end{aligned}$$

where $D = e(g,g)^{abc}$.

3) Randomly select a tuple $(w^*, \mathsf{W}^*) \in \mathcal{L}_{\tilde{G}}$, and calculates the solution of the received instance of the BCDH problem as follows:
$$D = \left(w^* \cdot e(g^c, V)^{1/x}\right)^{1/(x \cdot r)}.$$

Probability Analysis. Observe that if the simulator C does not abort the simulation, then it perfectly simulates the AUTH security experiment in the adversary $\mathcal{A}$'s view. Denote by q_{dk} and q_{sk} the total numbers of queries issued to $\mathcal{Q}_{\mathsf{RKeyGen}}(\cdot)$ and $\mathcal{Q}_{\mathsf{SKeyGen}}(\cdot)$ by $\mathcal{A}$, respectively. Then, from the analysis in [1] and [2], we know that the probability that C does not abort in the query phase is $\sigma^{q_{dk}+q_{sk}}$. Analogously, the probability that C does not abort in the forgery phase is $(1-\sigma)^2$. Therefore, if we denote by E_0 the even that C does not abort throughout the whole simulation, then we have that
$$\Pr[\mathsf{E}_0] = \sigma^{q_{dk}+q_{sk}} \cdot (1-\sigma)^2,$$
which is maximized at $\sigma' = (q_{dk} + q_{sk})/(q_{dk} + q_{sk} + 2)$. If we use σ' as the probability of C sampling 0 from $\{0, 1\}$ in the query phase, then we further have that
$$\Pr[\mathsf{E}_0] \geq \frac{4}{e^2 \cdot (q_{dk} + q_{sk} + 2)^2}.$$

Furthermore, if C makes correct guess on j^* and w^* (denote by this event E_1), then it gets a correct solution of the instance of the BCDH problem. The probability of E_1 is bounded as
$$\Pr[\mathsf{E}_1] \geq \frac{2 \cdot \mathsf{Adv}^{\text{AUTH}}_{\mathcal{A},\text{PIB-MKEM}}(\lambda, m, k)}{k \cdot q_{\tilde{G}}},$$
where $q_{\tilde{G}}$ is the number of queries to the random oracle $\tilde{G}(\cdot)$

Finally, the advantage of C correctly solving the instance of the BCDH problem is captured as follows:
$$\begin{aligned}\mathsf{Adv}^{\text{BCDH}}_{C}(\lambda) &= \Pr\left[C(g, g^a, g^b, g^c) = e(g,g)^{abc}\right] \\ &= \Pr[\mathsf{E}_0 \wedge \mathsf{E}_1] \\ &\geq \frac{8 \cdot \mathsf{Adv}^{\text{IND-MIS-CPA}}_{\mathcal{A},\text{PIB-MKEM}}(\lambda, m, k)}{e^2 \cdot k \cdot q_{\tilde{G}} \cdot (q_{dk} + q_{sk} + 2)^2}.\end{aligned}$$

This completes the proof. □

References

[1] D. Boneh and M. Franklin, "Identity-based encryption from the weil pairing," in *Advances in Cryptology–CRYPTO 2001*. Springer, 2001, pp. 213–229.

[2] G. Ateniese, D. Francati, D. Nuñez, and D. Venturi, "Match me if you can: Matchmaking encryption and its applications," in *Advances in Cryptology–CRYPTO 2019*. Springer, 2019, pp. 701–731.