Appendix A Security Results and Analysis

Appendix A.1 Security Result of 2ORKE

Theorem 1. Assume that the bit-size of $\mathcal{K}_{\mathsf{Eph}}$ is ν and the output length of CRHF is μ. Suppose that DPRF is a secure double pseudo-random function family, SIG is deterministic and SEUF-WCMA secure, and the hash function CRHF is collision-resistant, and the DDH assumption holds in $\mathbb{G}$ with prime order p. Then the proposed 2ORKE protocol is session-key-secure with $\mathsf{Adv}^{\mathsf{ake}}_{\mathsf{2ORKE},\mathcal{A}}(\lambda) \leqslant \frac{(d\ell)^2}{2^\nu} + \mathsf{Adv}^{\mathsf{cr}}_{\mathsf{CRHF},\mathcal{H}}(\lambda) + \ell \cdot \mathsf{Adv}^{\mathsf{seuf\text{-}wcma}}_{\mathsf{SIG},\mathcal{F}}(\lambda, d) + 4(d\ell)^2 \cdot (2 \cdot \mathsf{Adv}^{\mathsf{ind\text{-}cma}}_{\mathsf{DPRF},\mathcal{B}}(\lambda, d) + 2\mu \cdot \mathsf{Adv}^{\mathsf{ddh}}_{\mathbb{G},p,\mathcal{D}}(\lambda))$.

Proof. In this proof, we are going to reduce the security of our proposed protocol to that of underlying building blocks and the DDH problem. Let $\pi^{s^*}_{\mathsf{id}_i}$ denote the test oracle; and $\pi^{t^*}_{\mathsf{id}_j}$ be the origin oracle of $\pi^{s^*}_{\mathsf{id}_i}$. Note that the partner oracle of the test oracle is its origin oracle. Since each protocol message of our scheme includes the identity of sender, the origin oracle must come from the intended partner (i.e. id_j) of $\pi^{s^*}_{\mathsf{id}_i}$. Let Adv_ξ denote the advantage of $\mathcal{A}$ in the following Game ξ. Note that the adversary cannot query RevealKey to the test oracle or its partner oracle (if it exists). However, the adversary can ask different combinations of RevealEphKey and Corrupt queries to the test oracle and its origin oracle. We further list all freshness cases regarding RevealEphKey and Corrupt queries that $\mathcal{A}$ may ask. Then the following events and cases are obtained:

- **E1** : If $\pi^{z^*}_{\mathsf{id}_j}$ exists : (i) **C1** – $\mathcal{A}$ did not query $\mathsf{RevealEphKey}(\mathsf{id}_i, s^*)$ nor $\mathsf{RevealEphKey}(\mathsf{id}_j, z^*)$; (ii) **C2** : $\mathcal{A}$ did not query $\mathsf{Corrupt}(\mathsf{id}_i, pk^*)$ nor $\mathsf{RevealEphKey}(\mathsf{id}_j, z^*)$; (iii) **C3** : $\mathcal{A}$ did not query $\mathsf{Corrupt}(\mathsf{id}_i, pk^*)$ nor $\mathsf{Corrupt}(\mathsf{id}_j, pk^*)$; (iv) **C4** : $\mathcal{A}$ did not query $\mathsf{RevealEphKey}(\mathsf{id}_i, s^*)$ nor $\mathsf{Corrupt}(\mathsf{id}_j, pk^*)$.
- **E2** : If $\pi^{z^*}_{\mathsf{id}_j}$ does not exist : (v) **C5** : $\mathcal{A}$ did not query $\mathsf{Corrupt}(\mathsf{id}_i, pk^*)$ nor $\mathsf{Corrupt}(\mathsf{id}_j, pk^*)$ priori to the acceptance of $\pi^{s^*}_{\mathsf{id}_i}$; (vi) **C6** : $\mathcal{A}$ did not query $\mathsf{RevealEphKey}(\mathsf{id}_i, s^*)$ nor $\mathsf{Corrupt}(\mathsf{id}_j, pk^*)$ priori to the acceptance of $\pi^{s^*}_{\mathsf{id}_i}$.

In the following, we are going to show that our protocol is secure under all above freshness cases.

Game 0. The first game is the real security experiment, i.e. all queries in this game are simulated honestly in terms of protocol specification. Thus we have that $\mathsf{Adv}^{\mathsf{ake}}_{\mathsf{2ORKE},\mathcal{A}}(\lambda) = \mathsf{Adv}_0$.

Game 1. In this game, the challenger proceeds exactly like the previous game, except that it adds an additional abort rule. Namely the challenger aborts, if during the simulation an oracle generates an ephemeral public key which has appeared before. Note that if there are two oracles (e.g., $\pi^s_{\mathsf{id}_i}$ and $\pi^t_{\mathsf{id}_i}$) of a party which have collisions on ephemeral secret keys with non-negligible probability, then the adversary can ask $\mathsf{Corrupt}(\mathsf{id}_i)$ and ask $\mathsf{RevealEphKey}(\mathsf{id}_i, t)$ to learn the long-term key ss_{id_i} and $esk^t_{\mathsf{id}_i}$. This implies that the adversary learns all secrets of oracle $\pi^s_{\mathsf{id}_i}$ but without breaking the freshness of it. In this case, the adversary can select oracle $\pi^s_{\mathsf{id}_i}$ as test oracle and win the game. The above abort event is used to get rid of this situation. The randomness of ephemeral public key then is relative to that of ephemeral secret key which has length ν as assumed. Hence such abort event occurs with probability at least $\frac{(d\ell)^2}{2^\nu}$ in terms of birthday paradox. We have that $\mathsf{Adv}_0 \leqslant \mathsf{Adv}_1 + \frac{(d\ell)^2}{2^\nu}$. In this game, the ephemeral secret of DPRF in each oracle is unique. Thus, the exposed ephemeral secret key of an oracle would not affect other fresh oracles.

Game 2. This game proceeds as the previous game, except that the simulator aborts if: two oracles $\pi^s_{\mathsf{id}_i}$ and $\pi^t_{\mathsf{id}_j}$ accepted such that $\mathsf{CRHF}(\mathsf{sid}^s_{\mathsf{id}_i}) = \mathsf{CRHF}(\mathsf{sid}^t_{\mathsf{id}_j})$ but without matching sessions. When the event does occur, we can easily construct algorithm that breaks the collision-resistant hash function CRHF by outputting messages $(\mathsf{sid}^s_{\mathsf{id}_i}, \mathsf{sid}^t_{\mathsf{id}_j})$. Hence we have that $\mathsf{Adv}_1 \leqslant \mathsf{Adv}_2 + \mathsf{Adv}^{\mathsf{cr}}_{\mathsf{CRHF},\mathcal{H}}(\lambda)$.
This implies that each hashed session identifier is also uniquely shared by two partner oracles either.

Game 3. This game proceeds exactly as the previous game, but the challenger aborts if: there is an oracle $\pi^{s^*}_{\mathsf{id}_i}$ received a message $(\mathsf{id}_j, epk_{\mathsf{id}_j}, \sigma_{\mathsf{id}_j})$ which is not sent by any oracle of id_j before id_j is corrupt, but $\mathsf{SIG.Vfy}(vk^{sig}_{\mathsf{id}_j}, \sigma_{\mathsf{id}_j}, \mathsf{id}_j || epk_{\mathsf{id}_j}) = 1$. If the challenger aborts with overwhelming probability, then we could construct a signature forger $\mathcal{F}$. Meanwhile, $\mathcal{F}$ needs to guess the party that the adversary can forge. This would lose a factor of ℓ. We refer a reader to our previous work [**?**] for the detail of the security proof of this game (due to lacking of space). Therefore we have that $\mathsf{Adv}_2 \leqslant \mathsf{Adv}_3 + \ell \cdot \mathsf{Adv}^{\mathsf{seuf\text{-}erwcma}}_{\mathsf{SIG},\mathcal{F}}(\lambda, d)$.
As a result, the test oracle $\pi^{s^*}_{\mathsf{id}_i}$ in this game always has an origin oracle $\pi^{t^*}_{\mathsf{id}_j}$ before id_j is corrupted. This implies the freshness cases **C5** and **C6** never occur in this game.

Game 4. This game proceeds as before, but the challenger $\mathcal{C}$ tries to guess the test oracle and its origin oracle and one of the freshness cases. Technically, the challenger aborts if it fails in this guess. Since there are 4 fresh cases (i.e., from **C1** to **C4**) and ℓ parties at all, and at most d oracles for each party, then the probability of a correct guess is at least $1/4(d\ell)^2$. Thus we have that $\mathsf{Adv}_3 \leqslant 4(d\ell)^2 \cdot \mathsf{Adv}_4$.

Game 5. We change this game from the previous game by replacing the DPRF for the test oracle and its origin-oracle with two truly random functions respectively. If there exists an adversary $\mathcal{A}$ which can distinguish this game from the previous game, then there must exist another adversary $\mathcal{D}$ which can break the security of DPRF. Specifically, $\mathcal{D}$ could simulate the game for $\mathcal{A}$ as the challenger. In order to keep the freshness of the test oracle, $\mathcal{A}$ would only query either $\mathsf{RevealEphKey}(\mathsf{id}_i, s^*)$ or $\mathsf{Corrupt}(\mathsf{id}_i)$, then at least one key of DPRF is unexposed to $\mathcal{A}$. Note that $\mathcal{D}$ could choose one of the keys of DPRF (as specified in the security experiment of DPRF) that enables her to answer the RevealEphKey or Corrupt query correctly from $\mathcal{A}$.
For instance, $\mathcal{D}$ runs $\mathsf{Init}(\cdot)$ following its guess on freshness cases (as in the previous game), and generates the Diffie-Hellman exponents of the test oracle using its $\mathsf{RF}_{\mathsf{DPRF}}(\phi^{s^*}_{\mathsf{id}_i}, m^{s^*}_{\mathsf{id}_i})$ oracle query, where $m^{s^*}_{\mathsf{id}_i}$ is chosen at random by $\mathcal{D}$ and $\phi^{s^*}_{\mathsf{id}_i}$ is equivalent to the exposed key in the set $\{ss_{\mathsf{id}_i}, es^{s^*}_{\mathsf{id}_i}\}$. Similar simulation steps could be done to the origin-oracle. Meanwhile, the $\mathcal{D}$ may play two hybrid sub-games to change the DPRF progressively for the test oracle and its origin-oracle.

If $\mathsf{RF}_{\mathsf{DPRF}}$ is assigned with a truly random function, then the simulated game equals to this game, otherwise it is equivalent to the previous game. Due to the security of DPRF, we therefore have that $\mathsf{Adv}_4 \leqslant \mathsf{Adv}_5 + 2 \cdot \mathsf{Adv}^{\mathsf{ind\text{-}cma}}_{\mathsf{DPRF},\mathcal{B}}(\lambda, d)$. In this game, both the $esk^{s^*}_{\mathsf{id}_i}$ and $esk^{t^*}_{\mathsf{id}_j}$ (of the test oracle and its origin oracle respectively) are hidden from the adversary.

Game 6. In this game, we replace the session key of the test oracle and its partner oracle (if it exists) with a random value. If the $\mathcal{A}$ is able to distinguish this game from the previous game, then we can make use of $\mathcal{A}$ to build a DDH solver $\mathcal{D}$ as follows. On input a DDH challenge (g, p, g^a, g^b, g^c), the goal of $\mathcal{D}$ is to distinguish whether $g^c = g^{ab}$. $\mathcal{D}$ then simulates the AKE challenger for $\mathcal{A}$. Recall that only two partnered oracles would have the same hashed session identifier $h = \mathsf{CRHF}(\mathsf{sid})$ due to the results of previous games. The ephemeral secret key and public key (used to compute the session key) are programmed by the 'bits' $(h(1), \ldots, h(\mu))$ of such hashed session identifier. If the test oracle has no partner oracle but only origin oracle, then the hashed session identifier $h^{s^*}_{\mathsf{id}_i}$ of the test oracle $\pi^{s^*}_{\mathsf{id}_i}$ must be distinct to that $h^{t^*}_{\mathsf{id}_j}$ of its origin oracle $\pi^{t^*}_{\mathsf{id}_j}$, i.e., $h^{s^*}_{\mathsf{id}_i} \neq h^{t^*}_{\mathsf{id}_j}$. Since $h^{s^*}_{\mathsf{id}_i} \neq h^{t^*}_{\mathsf{id}_j}$, there must exist one bit, with coordinate τ^* in $h^{t^*}_{\mathsf{id}_j}$, is distinct to the τ^*-th bit of $h^{s^*}_{\mathsf{id}_i}$. Hence $\mathcal{D}$ should guess the coordinate τ^* and the value of $h^{s^*}_{\mathsf{id}_i}(\tau^*)$, which has a successful probability at least $\frac{1}{2\mu}$. If $\mathcal{D}$ guessed incorrectly, then it aborts. Otherwise $\mathcal{D}$ sets ephemeral keys $X^{s^*}_{\mathsf{id}_i,\tau^*,h^{s^*}_{\mathsf{id}_i}(\tau^*)} := g^a$ and $X^{t^*}_{\mathsf{id}_j,\tau^*,h^{s^*}_{\mathsf{id}_i}(\tau^*)} := g^b$, respectively. All other Diffie-Hellman keys are generated in terms of protocol specification using the ephemeral secret keys chosen by $\mathcal{D}$. This particularly implies that $\mathcal{D}$ knows almost all ephemeral secret keys except for $x^{s^*}_{\mathsf{id}_i,\tau^*,h^{s^*}_{\mathsf{id}_i}(\tau^*)}$ and $x^{t^*}_{\mathsf{id}_j,\tau^*,h^{s^*}_{\mathsf{id}_i}(\tau^*)}$.

We change the session key generation (in the view of the test oracle) for the test oracle and its partner oracle as the equation (A1).

$$
\begin{aligned}
k^* &= \Big(\prod_{\eta=1}^{\mu} X^{t^*}_{\mathsf{id}_j,\eta,h^{s^*}_{\mathsf{id}_i}(\eta)}\Big)^{\sum_{\iota=1,\iota\neq\tau^*}^{\mu} x^{s^*}_{\mathsf{id}_i,\iota,h^{s^*}_{\mathsf{id}_i}(\iota)}} \cdot \Big(\prod_{\eta=1}^{\mu} X^{t^*}_{\eta,h^{s^*}_{\mathsf{id}_i}(\eta)}\Big)^{x^{s^*}_{\mathsf{id}_i,\tau^*,h^{s^*}_{\mathsf{id}_i}(\tau^*)}} \qquad \text{(A1)}\\
&= \Big(\prod_{\eta=1}^{\mu} X^{t^*}_{\mathsf{id}_j,\eta,h^{s^*}_{\mathsf{id}_i}(\eta)}\Big)^{\sum_{\iota=1,\iota\neq\tau^*}^{\mu} x^{s^*}_{\mathsf{id}_i,\iota,h^{s^*}_{\mathsf{id}_i}(\iota)}} \cdot \Big(\prod_{\eta=1}^{\mu} X^{t^*}_{\mathsf{id}_j,\eta,h^{s^*}_{\mathsf{id}_i}(\eta)}\Big)^{x^{s^*}_{\mathsf{id}_i,\tau^*,h^{s^*}_{\mathsf{id}_i}(\tau^*)}}\\
&\cdot \Big(\prod_{\eta=1,\eta\neq\tau^*}^{\mu} X^{t^*}_{\mathsf{id}_j,\eta,h^{s^*}_{\mathsf{id}_i}(\eta)}\Big)^{x^{s^*}_{\mathsf{id}_i,\tau^*,h^{s^*}_{\mathsf{id}_i}(\tau^*)}} \cdot \big(X^{t^*}_{\mathsf{id}_j,\tau^*,h^{s^*}_{\mathsf{id}_i}(\tau^*)}\big)^{x^{s^*}_{\mathsf{id}_i,\tau^*,h^{s^*}_{\mathsf{id}_i}(\tau^*)}}.
\end{aligned}
$$

We need to do the above modification because we want to split the session key generation of the test oracle into two parts. The first part is the value that the adversary $\mathcal{D}$ is able to compute using the ephemeral secrets of her own choice. The second part is the one, i.e. $\big(X^{t^*}_{\mathsf{id}_j,\tau^*,h^{s^*}_{\mathsf{id}_i}(\tau^*)}\big)^{x^{s^*}_{\mathsf{id}_i,\tau^*,h^{s^*}_{\mathsf{id}_i}(\tau^*)}}$, which cannot by computed by $\mathcal{D}$, but will be replaced by the challenge value. Specifically, the session key of the test oracle and its partner oracle is further changed as the equation (A2):

$$
\tilde{k^*} := g^c \cdot \Big(\prod_{\eta=1}^{\mu} X^{t^*}_{\mathsf{id}_j,\eta,h^{s^*}_{\mathsf{id}_i}(\eta)}\Big)^{\sum_{\iota=1,\iota\neq\tau^*}^{\mu} x^{s^*}_{\mathsf{id}_i,\iota,h(\iota)}} \cdot \Big(\prod_{\eta=1}^{\mu} X^{t^*}_{\mathsf{id}_j,\eta,h^{s^*}_{\mathsf{id}_i}(\eta)}\Big)^{x^{s^*}_{\mathsf{id}_i,\tau^*,h^{s^*}_{\mathsf{id}_i}(\tau^*)}} \cdot \Big(\prod_{\eta=1,\eta\neq\tau^*}^{\mu} X^{t^*}_{\mathsf{id}_j,\eta,h^{s^*}_{\mathsf{id}_i}(\eta)}\Big)^{x^{s^*}_{\mathsf{id}_i,\tau^*,h^{s^*}_{\mathsf{id}_i}(\tau^*)}}. \qquad \text{(A2)}
$$

As for the origin oracle which is not the partner oracle of the test oracle, $\mathcal{D}$ can compute it using the ephemeral secrets chosen by herself, i.e. $x^{t^*}_{\mathsf{id}_j,1,h^{t^*}_{\mathsf{id}_j}(1)}, \ldots, x^{t^*}_{\mathsf{id}_j,\mu,h^{t^*}_{\mathsf{id}_j}(\mu)}$.

The simulation of $\mathcal{D}$ is perfect so far. If $g^c = g^{ab}$ then the game is equivalent to the previous game, otherwise the game is identical to this game. Due to the hardness of DDH problem, we therefore obtain that $\mathsf{Adv}_5 \leqslant \mathsf{Adv}_6 + 2\mu \cdot \mathsf{Adv}^{\mathsf{ddh}}_{\mathbb{G},p,\mathcal{D}}(\lambda)$.

Note that in this game the session key returned by the Test query is totally a truly random value which is independent of the bit b chosen by the Test query and any messages. Thus, the advantage that the adversary wins in this game is $\mathsf{Adv}_6 = 0$. Put altogether the probabilities from Game 0 to Game 6, we obtained the overall result of this theorem.

Appendix A.2 Security Result of MORKE

Theorem 2. Assume that the bit-size of $\mathcal{K}_{\mathsf{Eph}}$ is ν the output length of CRHF is μ. Suppose that DPRF is a secure double pseudo-random function family, SIG is deterministic and SEUF-WCMA secure, and CRHF is collision-resistant, and

the n-multilinear Diffie-Hellman assumption relative to MGen holds. Then the proposed MORKE protocol is session-key-secure with $\mathsf{Adv}^{\mathsf{ake}}_{\mathsf{MORKE},\mathcal{A}}(\lambda) \leqslant \frac{(d\ell)^2}{2^{\nu-1}} + \mathsf{Adv}^{\mathsf{cr}}_{\mathsf{CRHF},\mathcal{H}}(\lambda) + \ell \cdot \mathsf{Adv}^{\mathsf{seuf\text{-}wcma}}_{\mathsf{SIG},\mathcal{F}}(\lambda, d) + (2d\ell)^{n+1} \cdot ((n+1)\mathsf{Adv}^{\mathsf{ind\text{-}cma}}_{\mathsf{DPRF},\mathcal{B}}(\lambda, d) + (4\mu)^n \cdot \mathsf{Adv}^{\mathsf{mddh}}_{\mathsf{MGen},\mathcal{D}}(\lambda, n-1))$.

Proof. The proof of this theorem is quite similar to the proof of Theorem 1. During the reduction, the challenger may need to guess the test oracle and its origin oracle and one of freshness cases related to $\mathsf{RevealEphKey}$ and $\mathsf{RevealEphKey}$ queries. Since there are 2^{n+1} such fresh cases and ℓ parties at all, and at most d oracles for each party, then the guess would lead to loss a factor at least $(2d\ell)^{n+1}$. We refer the reader to [**?**, **?**] for further details about the number of freshness cases. In the following, we put emphasize on the reduction to the MDDH problem. One could refer to the other proofs to that of theorem 1.

In the security games, when we modify the game by replacing the session key of the test oracle $\pi^{s^*}_{\mathsf{id}_i}$ and its partner oracle (if it exists) with a random value. If the $\mathcal{A}$ is able to distinguish this change, then we can construct a MDDH distinguisher by using $\mathcal{A}$ as follows. On input a MDDH challenge instance $(\mathcal{PG}, g^{a_1}, \ldots, g^{a_n}, \Gamma)$, the goal of $\mathcal{D}$ is to distinguish whether or not $\Gamma = me(g^{a_1 \cdots a_n}, \ldots, g)$. $\mathcal{D}$ then simulates the AKE challenger for $\mathcal{A}$. Again only two partnered oracles would have the same hashed session identifier $h = \mathsf{CRHF}(\mathsf{sid})$ due to the security of corresponding cryptographic primitives.

We focus on the situation that the test oracle has no partner oracle but only origin oracles. Note that there are at most n origin oracles. In this case, $\mathcal{D}$ needs to do the following guesses before ephemeral key generation: (i) the $\tau^{t^*}_{\mathsf{id}_j}$-th bit of $h^{t^*}_{\mathsf{id}_j}$ (of an origin oracle $\pi^{t^*}_{\mathsf{id}_j}$) which is distinct to the τ^*-th bit of $h^{s^*}_{\mathsf{id}_i}$ (of the test oracle); (ii) the values $h^{s^*}_{\mathsf{id}_i}(\tau^*)$ and $h^{t^*}_{\mathsf{id}_j}(\tau^*)$. The $\mathcal{D}$ has to guess the above issues for all n origin oracles respectively. The success probability of above guessing is at least $\frac{1}{(4\mu)^n}$. We assume that $\mathcal{D}$ guessed correctly otherwise it aborts. Next, $\mathcal{D}$ sets ephemeral keys $X^{s^*}_{\mathsf{id}_i, \tau^{s^*}_{\mathsf{id}_i}, h^{s^*}_{\mathsf{id}_i}(\tau^{s^*}_{\mathsf{id}_i})} := g^{a_1}$ and all n challenge values to the n ephemeral keys $(X^{s^*}_{\mathsf{id}_i, \tau^{t^*}_{\mathsf{id}_j}, h^{s^*}_{\mathsf{id}_i}(\tau^{t^*}_{\mathsf{id}_j})})$ of n origin oracles respectively where $\tau^{t^*}_{\mathsf{id}_j}$ is the coordinate guessed before for specific origin oracle. $\mathcal{D}$ simulates all other Diffie-Hellman keys using the ephemeral secret keys chosen by herself.

Let the $((epk_1, esk_1), \ldots, (epk_n, esk_n))$ denote the ephemeral key pairs which are used to compute the session key of the test oracle, where $epk_i = (X_{i,\iota,\eta})_{(\iota,\eta)\in[\mu]\times\{0,1\}}$. Let the hashed session identifier of the test oracle is $h = (h(1), \ldots, h(\mu))$. Note that, in each epk_i, there is at least one Diffie-Hellman key is set as challenge value. We let τ_i denote the coordinate such that $X_{i,\tau_i,h(\tau_i)}$ is set to be challenge value. Let $Y_i = X_{i,\tau_i,h(\tau_i)}$ and $Z_i = \prod_{j=1,j\neq\tau_i}^{\mu} X_{i,j,h(j)}$. This implies the ephemeral secret key of Y_i is $y_i = x_{i,\tau_i,h(\tau_i)}$ and the ephemeral secret key of Z_i is $z_i = \sum_{j=1,j\neq\tau_i}^{\mu} x_{i,j,h(j)}$. Now we can rewrite the equation of the session key generation of the test oracle and its partner oracle as: $k^* = me(Y_2Z_2, Y_3Z_3, \ldots, Y_{n+1}Z_{n+1})^{y_1+z_1}$.

Let $\alpha = me(Y_2Z_2, Y_3Z_3, \ldots, Y_{n+1}Z_{n+1})^{z_1}$ and $\Delta = me(Y_2Z_2, Y_3Z_3, \ldots, Y_{n+1}, Z_{n+1})^{y_1}$. Then we have $k^* = \alpha \cdot \Delta$. It is not hard to see that α is computable by $\mathcal{D}$. Now we rewrite the equation for computing Δ as the equation (A3):

$$
\begin{aligned}
\Delta :=& me(Y_1^{z_2}, Y_3Z_3, \ldots, Y_nZ_{n+1}) \cdot me(Y_2, Y_3Z_3, \ldots, Y_{n+1}Z_{n+1})^{y_1} \\
=& me(Y_1^{z_2}, Y_3Z_3, \ldots, Y_{n+1}Z_{n+1}) \cdot me(W_2, Y_1^{z_3}, Y_4Z_4, \ldots, Y_{n+1}Z_{n+1}) \\
& \cdot me(W_2, W_3, Y_1^{z_4}, Y_5Z_5, \ldots, Y_{n+1}Z_{n+1}) \cdot \\
& \ldots \\
& \cdot me(Y_2, Y_3, Y_4, \ldots, Y_{n-1}, Y_1^{z_n}, Y_{n+1}Z_{n+1}) \cdot me(Y_2, Y_3, Y_4, \ldots, Y_{n-1}, Y_n, Y_1^{z_{n+1}}) \\
& \cdot me(Y_2, Y_3, \ldots, Y_{n+1})^{y_1}.
\end{aligned} \tag{A3}
$$

$\mathcal{D}$ replaces the value $me(Y_2, Y_3, \ldots, Y_{n+1})^{y_1}$ in the above equation with Γ to obtain a new Δ'. Then the session key of the test oracle and its partner oracle is computed $k^* = \alpha \cdot \Delta'$. As for the origin oracle which is not the partner oracle of the test oracle, $\mathcal{D}$ can compute it using the ephemeral secrets chosen by herself. The simulation of $\mathcal{D}$ is sound so far. If $\Gamma = me(g, g)^{a_1 \cdots a_n}$ then the game is equivalent to unchanged game, otherwise the game is identical to this one. The $\mathcal{D}$ could return what $\mathcal{A}$ outputs.