An Efficient Adaptive-Deniable-Concurrent Non-malleable Commitment Scheme

Seiko Arita

Graduate School of Information Security,
Institute of Information Security, Japan

Abstract. *It is known that composable secure commitments, that is, concurrent non-malleable commitments exist in the plaintext model, based only on standard assumptions such as the existence of claw-free permutations or even one-way functions. Since being based on the plaintext model, the deniability of them is trivially satisfied, and especially the latter scheme satisfies also adaptivity, hence it is* adaptive-deniable-concurrent non-malleable. *However, those schemes cannot be said to be practically efficient. We show a practically efficient (string) adaptive-deniable-concurrent commitment schemes is possible under a global setup model, called a global CRS-KR model.*

keywords: commitment schemes, adaptivity, deniability, concurrency, non-malleability.

1 Introduction

As advanced security properties, there are emerging concerns on composability, deniability and adaptivity of cryptographic protocols. The composability, culminating in Universal Composability (UC) [3], requires secure protocols remain secure even if they are composed with another protocols. The deniability requires secure protocols leave no evidence of their executions. The adaptivity requires secure protocols remain secure (to some reasonable extent) even if some honest parties are corrupted on way of their executions.

In this paper, we want to construct practically efficient commitment protocols with those advanced security properties in some global setup model.

It is known that composable secure commitments, that is, concurrent non-malleable commitments exist in the plaintext model, based only on standard assumptions such as the existence of claw-free permutations [21, 19] or even one-way functions [17]. Since being based on the plaintext model, the deniability of them is trivially satisfied, and especially the scheme of [19] satisfies also adaptivity, hence it is already an *adaptive-deniable-concurrent non-malleable* commitment scheme. However, those schemes cannot be said to be practically efficient.

We show a practically efficient (string) adaptive-deniable-concurrent non-malleable commitment scheme is possible under a global setup model, called a global CRS-KR model.

- We define a notion of adaptive-deniable-concurrent non-malleable commitments, that captures the three advanced properties all at once for commitment schemes in the global CRS-KR model.
- We define, as a more-easy-to-prove property, a straight-line equivocal-extractability of commitment schemes and prove that it (with some other auxiliary properties) yields the adaptive-deniable-concurrent non-malleability in the global CRS-KR model.
- We construct a straight-line equivocal-extractable commitment scheme in the global CRS-KR model, under the decisional linear assumption and the knowledge of exponent assumption on bilinear groups. The scheme is efficient and practical, using a constant number of pairing computations and three-round exchanges of linear-size messages.

Related works. In the literature there exists only one (to our knowledge) practically-efficient commitment scheme by Canetti, Dodis, Pass and Walfish [4], that establishes the adaptive-deniable-concurrent non-malleability. Their scheme relies on a global setup model, called augmented CRS model (a kind of global CRS model with an augmented functionality) and is proved to be adaptively UC-secure in that model. It uses four-round message exchanges of square size $O(k^2)$ of security parameter k. We believe that the global CRS-KR model that we use is arguably more simple than the augmented CRS model and our scheme, that uses three-round exchanges of messages of linear size $O(k)$, is more efficient than their scheme.

2 An Adaptive-Deniable-Concurrent Non-malleable Commitment

2.1 Commitment Schemes

In this paper, we deal with commitments in a tag-based manner. On input value v and tag t, sender $\mathcal{S}$ commits to v through a transcript c for receiver $\mathcal{R}$ using a CRS σ (Commitment phase). Later, using a local output d of the commitment phase, $\mathcal{S}$ decommits c to the value v using the same tag t and CRS σ (Decommitment phase). Hiding property requires even adversarial receiver $\mathcal{R}^*$ cannot know the value v under the commitment c in the commitment phase. Binding property requires even adversarial sender $\mathcal{S}^*$ cannot decommit a same commitment c to two different values. More precisely,

Definition 1 (Commitment Schemes). *A triple $\Sigma = (\mathcal{K}, \mathcal{S}, \mathcal{R})$ of probabilistic polynomial time algorithms is called a* (tag-based) commitment scheme *if it satisfies the following three properties:*

- *(Correctness) For any value $v \in \{0,1\}^k$ and any tag, the probability*

$$\Pr[\sigma \leftarrow \mathcal{K}(1^k),\ (d, c) \leftarrow \langle \mathcal{S}_{\sigma,tag}(v), \mathcal{R}_{\sigma,tag} \rangle,\ (-, v') \leftarrow \langle \mathcal{S}_{\sigma,tag}(d), \mathcal{R}_{\sigma,tag}(c) \rangle\ :\ v = v']$$

  is negligibly close to 1 (with respect to k).
- *(Hiding) For any adversary $\mathcal{R}^*$ and any tag, the probability*

$$\Pr[\sigma \leftarrow \mathcal{K}(1^k),\ (v_0, v_1, s) \leftarrow \mathcal{R}^*(\sigma, tag),\ b \xleftarrow{\$} \{0,1\},\ (-, b') \leftarrow \langle \mathcal{S}_{\sigma,tag}(v_b), \mathcal{R}^*(s) \rangle\ :\ b = b']$$

  is negligibly close to 1/2 (with respect to k). Here, v_0, v_1 are supposed to be in $\{0,1\}^k$.
- *(Binding) For any adversary $\mathcal{S}^*$ and any tag, the probability*

$$\Pr[\sigma \leftarrow \mathcal{K}(1^k),\ ((d_1, d_2), c) \leftarrow \langle \mathcal{S}^*(\sigma, tag), \mathcal{R}_{\sigma,tag} \rangle,\ (-, v_1) \leftarrow \langle \mathcal{S}^*(d_1), \mathcal{R}_{\sigma,tag}(c) \rangle,$$
$$(-, v_2) \leftarrow \langle \mathcal{S}^*(d_2), \mathcal{R}_{\sigma,tag}(c) \rangle\ :\ v_1 \neq \perp,\ v_2 \neq \perp,\ v_1 \neq v_2]$$

  is negligible (with respect to k).

For our purpose, we need a new type of binding property, *determining property.*

Definition 2 (Determining Property). *A commitment scheme $\Sigma = (\mathcal{K}, \mathcal{S}, \mathcal{R})$ is said to be* determining *if there exists a deterministic function* $\det(\cdot)$ *(that may not be efficiently computable) and for any feasible adversary $\mathcal{S}^*$ the probability*

$$\Pr[\sigma \leftarrow \mathcal{K}(1^k),\ (-, c) \leftarrow \langle \mathcal{S}^*(\sigma), R_{\sigma,tag} \rangle\ :\ \exists d, r \text{ s.t. } (-, v) = \langle \mathcal{S}^*(d), R_{\sigma,tag}(c; r) \rangle\ \Rightarrow\ v = \det(c)]$$

is negligibly close to 1 (with respect to k). Here, tag is being chosen by $\mathcal{S}^*$. We call the unique value $v = \det(c)$ determining value *of c.*

In the above definition we note that the d and r are not supposed to be efficiently computable. The determining property means that any commitment c generated by feasible adversaries S^* must statistically determine the value that can be committed to under c. If Σ is statistically binding then it is determining, and if Σ is determining then it is computationally binding. In the rest of paper, we only consider commitment schemes that are determining.

2.2 Definition of Adaptive-Deniable-Concurrent Non-malleability

Intuitively, we call a commitment scheme adaptive-deniable-concurrent non-malleable if the scheme permits no fake commitments even by a man-in-the-middle adversary that is able to invoke any number of left and right parties concurrently to receive/make commitments and is able to adaptively corrupt any number of left parties that completes the commitment phases, and in addition if the scheme leaves no evidence of execution of the protocol.

Since the property seems to be unreachable in the plain model (as seen later), in the rest of paper, we rely on a global setup model, called the global CRS-KR model. In that model, participants must register their identities to receive their public/private key pairs and a global CRS in advance, that are need to execute protocols. (Secret keys are used only when participants want to yield the deniability.) More formally, we define two experiments, as to a commitment scheme Σ and an adversary A, as in Figure 1.

- **Experiment** $\mathsf{mim}_A^{\Sigma}(v_1, \cdots, v_m, z)$:
  1. Invoke A on input z. A outputs a string ID^* to register itself and tags $t_1, \ldots, t_m$ for m left interactions.
  2. Generate CRS σ and a public/private-key pair pk_{ID^*}, sk_{ID^*} for ID^* as specified by Σ.
  3. Receiving $\sigma, pk_{ID^*}, sk_{ID^*}$, A carries over a concurrent man-in-the-middle attack: Each of m left honest parties concurrently commits to v_i with tag t_i according to Σ through transcript c_i for A $(i = 1, \ldots, m)$ and at the same time A concurrently commits to some values with tags t_j^* (chosen by A) through transcripts c_j^* for each of n right honest parties $(j = 1, \ldots, n)$ with some polynomial $n = n(k)$. (If necessary, A can receive public keys pk_{ID} of any ID it wants.)
  4. The experiment outputs $(view_A, v_1^*, \ldots, v_n^*, d_1, \ldots, d_m)$. Here, $view_A$ denotes the view of A in the above attack, v_j^* denotes the determining value $\det(c_j^*)$ of the right commitments c_j^* $(j = 1, \ldots, n)$ and d_i denotes honest decommitment of the left commitments c_i (to v_i) $(i = 1, \ldots, m)$. However here, for each $j \in \{1, \ldots, n\}$, if $t_j^* = t_i$ with some i, then v_j^* be overwritten to $\perp$.
- **Experiment** $\mathsf{sta}_S^{\Sigma}(v_1, \cdots, v_m, z)$:
  1. Invoke S on input z. S outputs a string ID^* to register itself and tags $t_1, \ldots, t_m$.
  2. Generate CRS σ and a public/private-key pair pk_{ID^*}, sk_{ID^*} of ID^* as specified by Σ.
  3. Receiving $\sigma, pk_{ID^*}, sk_{ID^*}$, S concurrently commits to some values using tags t_i^* (chosen by S) according to Σ through transcripts c_i^* for each of n receivers with some polynomial $n = n(k)$ $(i = 1, \ldots, n)$ and outputs a simulated view $view_A$ of A. (If necessary, S can receive public keys pk_{ID} of any ID it wants.)
  4. S is given the values $v_1, \cdots, v_m$. Then, S outputs decommitment d_i (to v_i) that corresponds to the simulated commitments c_i in the simulated view $view_A$ $(i = 1, \ldots, m)$.
  5. The experiment outputs $(view_A, v_1^*, \ldots, v_n^*, d_1, \ldots, d_m)$. Here, v_j^* denotes the determining value $\det(c_j^*)$ of the right commitments c_j^* $(j = 1, \ldots, n)$.

Fig. 1. Experiments mim_A^{Σ} and sta_S^{Σ}.

Definition 3. *A commitment scheme $\Sigma = (K, \mathcal{S}, \mathcal{R})$ is said to be* adaptive-deniable-concurrent non-malleable in the global CRS-KR model *if for any feasible adversary A, there exists a feasible*

algorithm S and the ensemble $\left\{\mathsf{sta}_S^{\Sigma}(v_1,\cdots,v_m,z)\right\}_{v_1,\cdots,v_m,z}$ is computationally indistinguishable from the ensemble $\left\{\mathsf{mim}_A^{\Sigma}(v_1,\cdots,v_m,z)\right\}_{v_1,\cdots,v_m,z}$ with any polynomial $m = m(k)$, any values $v_1,\cdots,v_m$ $(\in\{0,1\}^k)$ and any string z.

Remark 1. In the above definition, adversary A is restricted to be a type of "selective-tag", namely, A is supposed to choose left tags $t_1,\ldots,t_n$ before it knows the CRS and A must use different tags t_j^* from t_i for his challenges. The restriction is not restrictive. In fact, a tag-based commitment scheme that is secure for selective-tag-type adversaries is easily transformed into an ordinal commitment scheme without tags that is secure for fully adaptive adversaries, by using a method of the CHK transformation [7], just as in the cases of encryption schemes. (Sender generates a key-pair of strong one-time signature. It uses the verification key as a tag and signs a transcript by the corresponding secret key.)

2.3 Definition of Straight-line Equivocal-Extractability

As a more-easy-to-prove sufficient condition for the adaptive-deniable-concurrent non-malleability, we define straight-line equivocal-extractability of commitment schemes, that involves only a "classical" (instead of concurrent) man-in-the-middle adversary. The straight-line equivocal-extractability requires that a left party can be adaptively simulated by some feasible algorithm EQV that knows adversary's secret key and at the same time the property requires that the value committed to by the adversary for a right party can be extracted by some feasible algorithm EXT that knows the trapdoor of the global CRS. More formally, we define two experiments, as to a commitment scheme Σ and an adversary A, as in Figure 2.

Definition 4. *A commitment scheme $\Sigma = (K,\mathcal{S},\mathcal{R})$ is said to be* straight-line equivocal-extractable in the global CRS-KR model *if there exists a feasible algorithm EQV such that for any feasible adversary A, there exists a feasible algorithm EXT and the ensemble $\left\{\mathsf{fakeCom}_{\mathsf{EQV},A,\mathsf{EXT}}^{\Sigma}(v,z)\right\}_{v,z}$ is computationally indistinguishable from the ensemble $\left\{\mathsf{realCom}_A^{\Sigma}(v,z)\right\}_{v,z}$ with any value v $(\in\{0,1\}^k)$ and any string z.*

Remark 2. The definition requires that the algorithm EQV is independent of adversaries A.

Theorem 1. *If a commitment scheme $\Sigma = (K,\mathcal{S},\mathcal{R})$ is determining, public-coin for $\mathcal{R}$ and straight-line equivocal-extractable in the global CRS-KR model, then Σ is adaptive-deniable-concurrent non-malleable in the global CRS-KR model.*

Proof Idea. To prove the theorem, we construct a stand-alone man-in-the-middle adversary A_{ij} from an assumed concurrent adversary A against the scheme, that internally simulates left parties besides the i-th left party and right parties besides the j-th right party for A. We use the straight-line equivocal-extractability for A_{ij} to upper bound advantage of the assumed A, using a hybrid argument. In doing that, as one subtle point, we need to efficiently construct the view of $A_{i2},\ldots,A_{im}$ given the view of A_{i1}, that will be possible since the scheme is public-coin for receiver $\mathcal{R}$.

Proof. Let $\Sigma = (K,\mathcal{S},\mathcal{R})$ be a commitment scheme that is determining, public-coin for $\mathcal{R}$ and straight-line equivocal-extractable. Let EQV be the algorithm that is guaranteed to exist by the definition of straight-line equivocal-extractability for Σ. Let A be arbitrary feasible adversary against

– **Experiment** $\mathsf{realCom}_A^{\Sigma}(v, z)$:
  1. Invoke A on input z. A outputs a string ID^* to register itself and a tag t for the left interaction.
  2. Generate CRS σ and a public/private-key pair pk_{ID^*}, sk_{ID^*} of ID^* as specified by Σ.
  3. Receiving $\sigma, pk_{ID^*}, sk_{ID^*}$, A carries over a (classical) man-in-the-middle attack: A left honest party commits to v with tag t according to Σ through transcript c for A and at the same time A commits to some value using tag t^* (chosen by A) through transcript c^* for a right honest party. (If necessary, A can receive public keys pk_{ID} of any ID it wants.)
  4. The experiment outputs $(\tau, view_A, v^*, d)$. Here, τ denotes the trapdoor of CRS σ, $view_A$ denotes the view of A in the above attack, v^* denotes the determining value $\det(c^*)$ of the right commitment c^* and d denotes honest decommitment of the left commitment c (to v). However here, if $t^* = t$ then v^* be overwritten to $\perp$.

– **Experiment** $\mathsf{fakeCom}^{\Sigma}_{\mathsf{EQV},A,\mathsf{EXT}}(v, z)$:
  1. Invoke A on input z. A outputs a string ID^* to register itself and a tag t for the left interaction.
  2. Generate CRS σ with trapdoor τ and a public/private-key pair pk_{ID^*}, sk_{ID^*} of ID^* as specified by Σ.
  3. Receiving $\sigma, pk_{ID^*}, sk_{ID^*}$, A carries over a (classical) man-in-the-middle attack with a dummy left party: The algorithm EQV, on input t, sk_{ID}, plays the role of honest left party and commits to a dummy value w with tag t according to Σ through transcript c for A and at the same time A commits to some value using tag t^* (chosen by A) through transcript c^* for a right honest party. (If necessary, A can receive public keys pk_{ID} of any ID it wants.) Let d' be a local output of $\mathsf{EQV}(t, sk_{ID}, w)$.
  4. If $t^* = t$ then set v^* to be $\perp$. Otherwise, invoke algorithm EXT on input the trapdoor τ of CRS σ and the view $view_A$ of A, and set its output to v^*.
  5. Invoke (the second stage of) algorithm EQV on input d' and v, and get its output d.
  6. The experiment outputs $(\tau, view_A, v^*, d)$.

Fig. 2. Experiments $\mathsf{realCom}_A^{\Sigma}$ and $\mathsf{fakeCom}^{\Sigma}_{\mathsf{EQV},A,\mathsf{EXT}}$.

Σ. We can assume A is deterministic without loss of generality (by supposing A's coins are included in its input z). We want to construct a simulator S that satisfies

$$\mathsf{sta}_S^{\Sigma}(v_1, \cdots, v_m, z) \equiv_c \mathsf{mim}_A^{\Sigma}(v_1, \cdots, v_m, z) \tag{1}$$

for any values $v_1, \cdots, v_m$ ($\in \{0,1\}^k$) and any string z. We construct such simulator S as follows:

– **Simulator** S on input a string z:
  1. S invokes A on input z. A outputs $ID^*, t_1, \ldots, t_m$. Then, S also outputs $ID^*, t_1, \ldots, t_m$.
  2. S receives CRS σ and a public/private-key pair pk_{ID^*}, sk_{ID^*}. Giving those $\sigma, pk_{ID^*}, sk_{ID^*}$ to A, S plays each of m left parties for A by invoking EQV on t_i, sk_{ID^*}, w (Here, w is any string such as 0^k) ($i = 1, \ldots, m$). S forwards messages of A to each of n right parties to its own outside right parties $\mathcal{R}_{\sigma,t_j^*}$ and forwards messages from $\mathcal{R}_{\sigma,t_j^*}$ to A as its j-th right party messages ($j = 1, \ldots, n$). Let d_i' be the local output of $\mathsf{EQV}(t_i, sk_{ID^*}, w)$ ($i = 1, \ldots, m$).
  3. On halting A, S outputs its view $view_A$.
  4. Receiving left values v_i, S invokes the corresponding second stage of EQV on d_i' and v_i to get its output d_i and outputs d_i as its own output ($i = 1, \ldots, m$).

Toward showing Equation (1) with the above S, we first construct a (classical) man-in-the-middle adversary $A_{i,j}$ using A as follows ($i = 0, \ldots, m-1$, $j = 1, \ldots, n$).

– **A man-in-the-middle adversary** $A_{i,j}$ on input $z, v_{i+2}, \ldots, v_m$:
  1. $A_{i,j}$ internally invokes A on input z. A outputs $ID^*, t_1, \ldots, t_m$. Then, $A_{i,j}$ outputs ID^*, t_{i+1}.

2. Receiving CRS σ and a public/private-key pair pk_{ID^*}, sk_{ID^*}, $A_{i,j}$ carries over a man-in-the-middle attack against its (outside) left party and right party, using the internally simulated A that is given those $\sigma, pk_{ID^*}, sk_{ID^*}$.
   (a) In the attack, $A_{i,j}$ internally simulates m left parties for A as follows.
      i. $A_{i,j}$ simulates each of the first i left parties by using α-th copy of EQV on t_α, sk_{ID^*}, w ($\alpha = 1, \ldots, i$). (w is any dummy string.)
      ii. $A_{i,j}$ forwards messages from its left party to A as messages of A's $(i+1)$-th left party, and forwards messages from A to its $(i+1)$-th left party to its left party.
      iii. $A_{i,j}$ simulates the rest of $m - i - 1$ left parties honestly to commit to v_β ($\beta = i+2, \ldots, m$).
   (b) At the same time $A_{i,j}$ internally simulates n right parties for A as follows.
      i. $A_{i,j}$ honestly simulates each of n right parties except the j-th right party for A.
      ii. $A_{i,j}$ forwards messages from its right party to A as messages of A's j-th right party, and forwards messages from A to A's j-th right party to its right party.

– **Experiment** $\mathsf{hyb}\text{-}i_A^\Sigma(v_1, \ldots, v_m, z)$:
1. Invoke A on input z, that outputs a string ID^* to register itself and tags $t_1, \ldots, t_m$ for left interactions.
2. Generate CRS σ with trapdoor τ and a public/private-key pair pk_{ID^*}, sk_{ID^*}, and give them to A.
3. A begins a concurrent man-in-the-middle attack. Then,
   (a) Simulate each of first i left parties by α-th copy of EQV on t_α, sk_{ID^*}, w ($\alpha = 1, \ldots, i$).
   (b) Simulate honestly each of the rest $m - i$ left parties to commit to v_β ($\beta = i+1, \ldots, m$).
   (c) Receiving these m left commitments $c_1, \ldots, c_m$ concurrently, A commits to some values with tags t_j^* (chosen by A) through transcripts c_j^* for each of n right honest parties concurrently ($j = 1, \ldots, n$) with some polynomial $n = n(k)$.

   Let d'_α be the local output of $\mathsf{EQV}(t_\alpha, sk_{ID}, w)$ for $\alpha = 1, \ldots, i$ and d'_β be the honest decommitment corresponding to c_β for $\beta = i+1, \ldots, m$.
4. Let $A_{i,j}$ be the man-in-the-middle adversary defined as above from this A. We note that the above attack of A can be viewed as the attack of $A_{i,j}$ on input $z, v_{i+2}, \ldots, v_m$ that stands between the $(i+1)$-th left party and the j-th right party. Let EQV and $\mathsf{EXT}^{(i,j)}$ be feasible algorithms for $A_{i,j}$ that are guaranteed to exist by straight-line equivocal-extractability of Σ. (Recall EQV does not depend on the adversary by definition.)
5. For $j = 1, \ldots, n$, if $t_j^* = t_i$, set $v_j^* = \perp$, else invoke $\mathsf{EXT}^{(i,j)}$ on $\tau, view_{A_{i,j}}$ and set v_j^* to its output.
6. For $i = 1, \ldots, m$, invoke EQV on d'_i and v, and set d_i to its output.
7. The experiment outputs $(view_A, v_1^*, \ldots, v_n^*, d_1, \ldots, d_m)$.

Fig. 3. Hybrid Experiments $\mathsf{hyb}\text{-}i_A^\Sigma$.

Using the above $A_{i,j}$, we define hybrid experiments $\mathsf{hyb}\text{–}i_A^\Sigma(v_1, \ldots, v_m, z)$ for $i = 0, 1, \ldots, m$ as in Figure 3. In the experiments, when $i = m$, all of the left parties are simulated by EQV and $\mathsf{hyb}\text{–}m_A^\Sigma$ is exactly distributed as sta_S^Σ:

$$\mathsf{hyb}\text{–}m_A^\Sigma(v_1, \ldots, v_m, z) \equiv \mathsf{sta}_S^\Sigma(v_1, \cdots, v_m, z). \tag{2}$$

On the other hand, when $i = 0$, all of the left parties are honestly simulated to commit to v_i. Moreover the straight-line equivocal-extractability of Σ means that $\det(c_j^*) = \mathsf{EXT}^{(i,j)}(\tau, view_{A_{i,j}})$ (with exception of negligible probability). Hence, we have

$$\mathsf{hyb}\text{–}0_A^\Sigma(v_1, \ldots, v_m, z) \equiv_s \mathsf{mim}_A^\Sigma(v_1, \cdots, v_m, z). \tag{3}$$

By Equations (2) and (3), Equation (1) follows from the next claim by a standard hybrid argument.

Claim. For $i = 0, \ldots, m-1$, we have $\mathsf{hyb}\text{-}i_A^{\Sigma}(v_1, \ldots, v_m, z) \equiv_c \mathsf{hyb}\text{-}(i+1)_A^{\Sigma}(v_1, \ldots, v_m, z)$.

Proof. We define two auxiliary experiments real_A^{Σ} and $\mathsf{ideal}_A^{\Sigma}$. The experiment $\mathsf{real}_A^{\Sigma}(v_1, \ldots, v_m, z)$ first constructs $B = A_{i,1}$. Then, it computes $(\tau, view_B, v_1^*, d_{i+1}) \leftarrow \mathsf{realCom}_B^{\Sigma}(v_{i+1}, (z, v_{i+2}, \ldots, v_m))$, and returns $(view_A, v_1^*, \ldots, v_n^*, d_1, \ldots, d_n) \leftarrow \mathsf{Expand}(\tau, view_B, v_1^*, d_{i+1}, (v_1, \ldots, v_m))$. The experiment $\mathsf{ideal}_A^{\Sigma}$ is the same as real_A^{Σ} except that it uses $\mathsf{fakeCom}_{\mathsf{EQV},B,\mathsf{EXT}}^{\Sigma}$, instead of $\mathsf{realCom}_B^{\Sigma}$, to compute $(\tau, view_B, v_1^*, d_{i+1})$.

In the above, the procedure Expand reconstructs view of $A_{i,j}$ from the view of $B = A_{i,1}$ and extracts values v_j^* committed to by A in right interactions using EXT with trapdoor τ. More precisely, Expand proceeds as follows:

- **Procedure** $\mathsf{Expand}(\tau, view_B, v_1^*, d_{i+1}, (v_1, \ldots, v_m))$:
  1. For $j = 2, \ldots, n$, do:
     (a) Construct $A_{i,j}$'s view $view_{i,j}$ from $view_B$. (Recall $B = A_{i,1}$ and Σ is public-coin with respect to receivers.)
     (b) $v_j^* \leftarrow \mathsf{EXT}^{(i,j)}(\tau, view_{i,j})$ with the algorithm $\mathsf{EXT}^{(i,j)}$ for $A_{i,j}$.
  2. For $\alpha = 1, \ldots, i, i+2, \ldots, m$, do:
     (a) Reconstruct from $view_B$ the local output d'_α of the simulated α-th left party. (Recall $B = A_{i,1}$ internally simulates the α-th left party.)
     (b) $d_\alpha \leftarrow \mathsf{EQV}(d'_\alpha, v_\alpha)$
  3. Return $(view_A, v_1^*, \ldots, v_n^*, d_1, \ldots, d_n)$

It is easy to see that the view of A (included in $view_B$) in $\mathsf{real}_A^{\Sigma}(v_1, \ldots, v_m, z)$ is exactly distributed as $view_A$ in $\mathsf{hyb}\text{-}i_A^{\Sigma}$. (In both cases, the first i left parties are simulated by EQV.) Then, by definition of Expand, we have

$$\mathsf{hyb}\text{-}i_A^{\Sigma}(v_1, \ldots, v_m, z) \equiv \mathsf{real}_A^{\Sigma}(v_1, \ldots, v_m, z). \tag{4}$$

Similarly, we have

$$\mathsf{hyb}\text{-}(i+1)_A^{\Sigma}(v_1, \ldots, v_m, z) \equiv \mathsf{ideal}_A^{\Sigma}(v_1, \ldots, v_m, z). \tag{5}$$

The straight-line equivocal-extractability of Σ against B means that the outputs of $\mathsf{realCom}_B^{\Sigma}$ and $\mathsf{fakeCom}_{\mathsf{EQV},B,\mathsf{EXT}}^{\Sigma}$ are indistinguishable and then, since Expand is an efficient procedure, $\mathsf{real}_A^{\Sigma}(v_1, \ldots, v_m, z)$ and $\mathsf{ideal}_A^{\Sigma}(v_1, \ldots, v_m, z)$ are indistinguishable. The claim follows by Equation (4) and (5). □

□

Remark 3. We can construct a deniable concurrent zero-knowledge argument if the adaptive-deniable-concurrent non-malleable commitment scheme once exists, by using the GMW protocol of graph 3-coloring instantiated with it. That means

- It is difficult (if not impossible) to construct adaptive-deniable-concurrent non-malleable commitment schemes only using the CRS model, since deniable concurrent zero-knowledge arguments are known to be difficult in the CRS model [20].
- An efficient construction of adaptive-deniable-concurrent non-malleable commitment scheme in the global CRS-KR model, that will be shown in the next section, gives an efficient deniable concurrent zero-knowledge argument in the global CRS-KR model. (We remark that the efficient concurrent zero-knowledge scheme of [11] in the auxiliary string is not deniable.)

3 A Construction of Straight-line Equivocal-Extractable Commitment

This section constructs a determining, public-coin for receivers, and straight-line equivocal-extractable commitment scheme in the global CRS-KR model, using bilinear groups. The scheme is practically efficient, using a constant number of pairing computations and three-round exchanges of linear-size messages.

3.1 Design Principle

The basic design of our commitment scheme follows the one of Damgård and Nielsen [13]. Generate a one-time commitment key of a base commitment using a coin-flipping protocol and then commit to a value by the base commitment with the generated one-time key. CRS is used in the coin-flipping.

However, in our scheme, the generated coins used to form a one-time commitment key are not opened to a receiver, remaining secret of a sender. To assure the coins are honestly generated and used to form one-time commitment key, the sender appends a non-interactive zero-knowledge argument for proving that honesty. (As seen later, the used NIZK argument can be constructed without using Cook reduction, depending on the property of the bilinear map.) More accurately, the argument proves that the sender formed the one-time commitment key honestly *or* the sender knows the receiver's secret key, that enables only simulator EQV to form a fake equivocal commitment.

The scheme "duplicates" some items in the sender messages so that extractor EXT can extract the generated coins (used to form a one-time commitment key) and values committed to by adversaries in the course of proving security with help of some KEA extractors.

3.2 Building Blocks

The Homomorphic Commitment. As the base commitment, we use the homomorphic commitment of Groth, Ostrovsky, Sahai [16]. The homomorphic commitment is built on a group G with bilinear map. The commitment to a value m is computed as $\mathrm{Com}(m; r, t) = (M_1 = g_1^m g_4^r, M_2 = g_2^m g_5^t, M_3 = g_3^m g_6^{r+t})$ with a tuple $(g_1, \ldots, g_6)$ of six elements of G as a commitment key.

A tuple $(g_1, \ldots, g_6)$ is called linear when there exist α and β that satisfy $g_1 = g_4^\alpha, g_2 = g_5^\beta, g_3 = g_6^{\alpha+\beta}$. A linear tuple $(g_1, \ldots, g_6)$ defines an equivocal (so perfectly hiding) commitment: $\mathrm{Com}(m_0; r_0, t_0) = \mathrm{Com}(m; r_0{+}\alpha(m_0{-}m), t_0{+}\beta(m_0{-}m))$. On a while, a non-linear tuple $(g_1, \ldots, g_6)$ defines perfectly binding commitment. The commitment (M_1, M_2, M_3) with a non-linear $(g_i = g^{c_i})$ determines its underlying value m through $g^m = (M_1^{1/c_4} M_2^{1/c_5} M_3^{-1/c_6})^{1/(c_1/c_4 + c_2/c_5 - c_3/c_6)}$.

The Perfect Non-interactive Zero-Knowledge Argument. To prove the above-mentioned honesty of generation of one-time commitment key, we incorporate the non-interactive zero-knowledge argument of Abe and Fehr [1] into our scheme in a following way.

Let CRS be a pair of elements g, g_c in a group G with bilinear map. A Common input to a prover and a verifier is a triple of $A, b, \tilde{g}$ and a statement to be proved is: "There exist a and s satisfying that $A = g^a g_c^s$ and $\tilde{g} = g^{ab}$." The proof is simply a single element $P = g^s$, that is verified as $e(A, g^b) = e(\tilde{g}, g)e(P, g_c^b)$. The NIZKA is perfectly simulated in zero-knowledge, if one knows e_c satisfying $g_c = g^{e_c}$, as $P = (A\tilde{g}^{-1/b})^{1/e_c}$. The NIZKA is computationally sound, for example, under the Diffie-Hellman inversion assumption in a following way. Given an opening (a, s) of $A = g^a g_c^s$, an opening c of $\tilde{g} = g^c$ and a convincing proof P, one can efficiently compute g^{1/e_c} as $g^{1/e_c} = (g^{-s}P)^{b/(ab-c)}$ if $ab \neq c$. Note that we need openings of A and $\tilde{g}$ to use the soundness, that would require some use of KEA extractors, as in [1].

3.3 The Commitment Scheme

We describe our commitment scheme. Our scheme $\Sigma = (\Sigma'|\Sigma')$ executes two parallel independent copies of subscheme $\Sigma' = (K, S, R)$ in the global CRS-KR model. Each of these two executions uses an independent global CRS and an independent public/private key pair. If both executions of Σ' output a same value m, the scheme Σ outputs that value m. (Otherwise, it outputs $\perp$.)

Generation of CRS: K selects random four elements e_c, e_x, d_x, e_y from $\mathbb{Z}_q$ and a random element g of G. K computes $g_c = g^{e_c}$, $g_x = g^{e_x}$, $h_x = g_x^{d_x}$, $g_y = g^{e_y}$ and outputs $\sigma = (g, g_c, g_x, h_x, g_y)$.

Commitment phase: Sender $\mathcal{S}$ commits to value m ($\in \mathbb{Z}_q$) using tag tag and CRS $\sigma = (g, g_c, h_c, g_x, h_x)$ for receiver $\mathcal{R}$ with registered public key g_{ID} ($= g^{e_{ID}}$), as follows:

1. $\mathcal{S}$ randomly chooses $a_1, \ldots, a_6, s_1, \ldots, s_6, b'_1, \ldots, b'_6$ from $\mathbb{Z}_q^*$ and $w_1, \ldots, w_6$ from $\mathbb{Z}_q$. For $i = 1, \ldots, 6$, $\mathcal{S}$ computes $A_i = g^{a_i} {g_c}^{s_i}$, $U_i = g^{w_i} {g_{ID}}^{b'_i}$. $\mathcal{S}$ sends $A_1, \ldots, A_6, U_1, \ldots, U_6$ to $\mathcal{R}$.
2. $\mathcal{R}$ randomly chooses $b_1, \ldots, b_6$ from $\mathbb{Z}_q^*$ and sends them to $\mathcal{S}$.
3. In response $\mathcal{S}$ computes, for $i = 1, \ldots, 6$, $b''_i = b_i/b'_i$, $c_i = a_i b''_i$, $g_i = g^{c_i}$, $h_i = {G_x}^{c_i}$, $P_i = g^{s_i}$, $Q_i = {g_y}^{s_i}$ with $G_x = g_x^{H(tag)} h_x$. Then, $\mathcal{S}$ selects random elements r, t from $\mathbb{Z}_q$ and computes $M_1 = {g_1}^m {g_4}^r$, $M_2 = {g_2}^m {g_5}^t$, $M_3 = {g_3}^m {g_6}^{r+t}$. $\mathcal{S}$ sends the following items to $\mathcal{R}$, $g_1, \ldots, g_6, h_1, \ldots, h_6, P_1, \ldots, P_6, Q_1, \ldots, Q_6, b'_1, \ldots, b'_6, w_1, \ldots, w_6, M_1, M_2, M_3$.
4. $\mathcal{R}$ checks, for $i = 1, \ldots, 6$, $\mathrm{DH}(g, G_x, g_i, h_i)$, $\mathrm{DH}(g, g_y, P_i, Q_i)$, $U_i = g^{w_i} {g_{ID}}^{b'_i}$, $e(A_i, g^{b''_i}) = e(g_i, g) e(P_i, {g_c}^{b''_i})$ with $b''_i = b_i/b'_i$. If any of them (for any i) is not true, $\mathcal{R}$ aborts.

Decommitment phase: $\mathcal{S}$ sends m, r, t to $\mathcal{R}$, who accepts it if $M_1 = {g_1}^m {g_4}^r$, $M_2 = {g_2}^m {g_5}^t$, $M_3 = {g_3}^m {g_6}^{r+t}$.

Fig. 4. The Commitment Subscheme $\Sigma' = (K, \mathcal{S}, \mathcal{R})$

Let G be a group of prime order q with bilinear map $e(\cdot, \cdot) : G \times G \to G_T$ and let H be an injective function from $\{0,1\}^{tagLen}$ to $\mathbb{Z}_q$ (with length $tagLen$ of tag strings). The subscheme $\Sigma' = (K, \mathcal{S}, \mathcal{R})$ on group G proceeds as in Figure 4. ($\mathrm{DH}(\cdot, \cdot, \cdot, \cdot)$ means that a given tuple constitutes a DH-tuple.) It relies on a global CRS consisting of five elements (g, g_c, g_x, h_x, g_y) in G and requires the knowledge of receiver's public-key g_{ID} that is also an element of G.

First, parties run a coin-tossing protocol using (part of) the CRS to generate random coins $c_1, \ldots, c_6$ in ${\mathbb{Z}_q}^*$. Then, sender $\mathcal{S}$ forms a one-time commitment key $(g_1, \ldots, g_6)$ using the generated coins $c_1, \ldots, c_6$ as $g_i = g^{c_i}$, and computes NIZK arguments $P_1, \ldots, P_6$ that yield the one-time key $(g_1, \ldots, g_6)$ was honestly generated from $c_1, \ldots, c_6$ or $\mathcal{S}$ knows the secret key of $\mathcal{R}$. (Construction of the OR proof follows the standard way of making OR-proof of Σ protocols [6].) In the course, some part of the CRS is used only after 'twisted' with the tag tag to prevent adversaries from copying generated coins from/to another sessions. Main body of commitment to input m is generated with homomorphic commitment using that one-time key as $M_1 = {g_1}^m {g_4}^r$, $M_2 = {g_2}^m {g_5}^t$, $M_3 = {g_3}^m {g_6}^{r+t}$.

In the scheme, several messages are duplicated such as $h_1, \ldots, h_6$ for $g_1, \ldots, g_6$ and $Q_1, \ldots, Q_6$ for $P_1, \ldots, P_6$. Those will be used by KEA-extractors to extract the coins generated by adversaries from its view in the proof of security. A decommitment is done in a canonical way.

As seen, the proposed scheme is practically efficient, using a constant number of pairing computations and three-round exchanges of linear-size messages.

3.4 Security

First, we review two necessary assumptions. Let G denote a group of k-bit prime order q. The *decisional linear assumption* holds on G if linear tuples $({g_4}^\alpha, {g_5}^\beta, {g_6}^{\alpha+\beta}, g_4, g_5, g_6)$ are computationally

indistinguishable from random tuples $(g_4{}^{\alpha}, g_5{}^{\beta}, g_6{}^{\gamma}, g_4, g_5, g_6)$ with random elements g_4, g_5, g_6 in G and random α, β, γ in $\mathbb{Z}_q$.

The knowledge of exponent assumption [10] means that it is possible only if one knows b to generate a pair (g^b, g^{ab}) given a random g^a. More formally, for feasible algorithms H, H^* and any string w, an experiment $\mathbf{Exp}^w_{G,H,H^*}$ is defined as follows. H is given $q, g, A = g^a$ (with random a from $\mathbb{Z}_q$) and w, and outputs (B, W). On the same inputs, H^* is invoked and outputs b. If $W = B^a$ and $B \neq g^b$, the experiment outputs 1, otherwise outputs 0. The *knowledge of exponent assumption* holds on G if for any w and any feasible adversary H (called KEA-adversary) there exists a feasible H^* (called KEA-extractor) with negligible advantage $\mathbf{Adv}^w_{G,H,H^*}(k) := \Pr[\mathbf{Exp}^w_{G,H,H^*}(k) = 1]$.

We begin to analyze security of the proposed commitment scheme $\Sigma = (\Sigma'|\Sigma')$. In the analysis, we use ordinal characters such as $a, b, c, \ldots$ for items in one of the two parallel executions of the subscheme Σ' on which we have a current focus, and use characters with superscript $+$ such as $a^+, b^+, c^+, \ldots$ for the corresponding items in the another parallel execution. First we see its computational hiding property.

Proposition 1. *Under the decisional linear assumption on G, the proposed commitment scheme is computationally hiding in the global CRS-KR model.*

Proof. Suppose that a feasible adversary $\mathcal{R}^*$ with identity ID^* breaks the hiding property of the proposed scheme Σ using tag tag with non-negligible probability.

Using such $\mathcal{R}^*$, we construct an efficient distinguisher D that distinguishes between a *pair* of linear tuples and a *pair* of random tuples. Given a pair of tuples as input, D simulates a sender for $\mathcal{R}^*$ and plugs the input tuple into the second sender-message as a one-time commitment key, for each of the two parallel executions of the subscheme Σ'. In doing that, D uses the knowledge of simulated secret key of $\mathcal{R}^*$, and proves it knows the secret key in the OR proof, instead of proving honesty of the fake one-time commitment key. More details follow.

Distinguisher D on input $((g_1, \ldots, g_6), (g_1^+, \ldots, g_6^+))$:

1. (Simulate a sender for $\mathcal{R}^*$.) D invokes $\mathcal{R}^*$ and does as follows, for each of the two parallel executions of the subscheme.
   (a) (Emulate CRS and a key-pair.) D selects random e_{ID^*} from ${\mathbb{Z}_q}^*$ and a random g from G. D sets $g_{ID^*} = g^{e_{ID^*}}$ and gives (g_{ID^*}, e_{ID^*}) as her pair of public/private keys to $\mathcal{R}^*$. In addition, D chooses random e_c, e_x, d_x, e_y from $\mathbb{Z}_q$, sets $g_c = g^{e_c},\ g_x = g^{e_x},\ h_x = g_x^{d_x},\ g_y = g^{e_y}$, and gives tag and (g, g_c, g_x, h_x, g_y) to $\mathcal{R}^*$. D records $\eta = e_x(H(tag) + d_x)$ for later use. (Note $G_x = g^{\eta}$.)
   (b) (Emulate the first message.) Receiving a pair of messages (m_0, m_1) from $\mathcal{R}^*$, D selects a random bit b (only this process is common for the two executions). Then, D selects random $(s_i), (k_i), (b_i'')$ from $\mathbb{Z}_q$ and computes $A_i = g_i^{1/b_i''} g_c^{s_i}$ (with g_i in the input) and $U_i = g^{k_i}$ for $i = 1, \ldots, 6$. D sends $(A_i), (U_i)$ to $\mathcal{R}^*$.
   (c) (Emulate the second message.) Receiving challenge (b_i^*) from $\mathcal{R}^*$, to generate the second sender-message $((g_i), (h_i), \ldots, (M_i))$, D uses its own input (g_i) (or (g_i^+)) as the (g_i) in the second message, computes (h_i) as $h_i = g_i^{\eta}$ with η recorded at Step 1a and honestly computes the proofs $P_i = g^{s_i},\ Q_i = {g_y}^{s_i}$. Then, with $b_i' = b_i^*/b_i''$, D computes $w_i = k_i - b_i' e_{ID^*}$ using the secret key e_{ID^*}. Finally, D computes (M_1, M_2, M_3) as honest homomorphic commitment to

m_b with its own input $(g_1, \ldots, g_6)$ using as a commitment key. D sends the second message $((g_i), (h_i), (P_i), (Q_i), (b'_i), (w_i), (M_i))$ to $\mathcal{R}^*$.

2. (Output.) If $\mathcal{R}^*$ outputs $\hat{b}$ which is equal to b, then D outputs 1, otherwise outputs 0.

We evaluate the probability that D outputs 1. When given both $(g_1, \ldots, g_6)$ and $(g_1^+, \ldots, g_6^+)$ are linear, the simulated view of $\mathcal{R}^*$ in D is completely independent of b by the perfect hiding property of homomorphic commitment with linear tuples. So, the probability that D outputs 1 is $1/2$.

Suppose both $(g_1, \ldots, g_6)$ and $(g_1^+, \ldots, g_6^+)$ are random tuples. Let c_i and a_i be values defined by $g_i = g^{c_i}$ and $a_i = c_i/b''_i$. As for transcripts generated by D, we have

$$
\begin{aligned}
A_i &= g_i^{1/b''_i} g_c^{s_i} = g^{c_i/b''_i} g_c^{s_i} = g^{a_i} g_c^{s_i} \\
U_i &= g^{k_i} = g^{w_i + b'_i e_{ID^*}} = g^{w_i} {g_{ID^*}}^{b'_i} \\
g_i &= g^{c_i},\ h_i = g_i^{\eta} = (g^{c_i})^{\eta} = (g^{\eta})^{c_i} = G_x^{c_i} \\
P_i &= g^{s_i},\ Q_i = {g_y}^{s_i}.
\end{aligned}
$$

Thus, the simulated transcript is also determined by values $(a_i), (s_i), (w_i), (b'_i), (c_i)$ just as the real transcripts. Here, by the description of D, we see $(c_i), (b''_i), (s_i), (k_i)$ among them are independently uniformly distributed and the rest $(a_i), (b'_i), (w_i)$ are determined through the relations $c_i = a_i b''_i,\ b^*_i = b'_i b''_i,\ w_i = k_i - b'_i x_{ID}$ by them. On a while, in the real transcript, $(a_i), (b'_i), (s_i), (w_i)$ are independently uniform and the remaining $(c_i), (b''_i), (k_i)$ are determined by the same relations by them. So, we see the distribution of $(a_i), (s_i), (w_i), (b'_i), (c_i)$ is identical regardless whether it is simulated or real. Hence, the simulated view of $\mathcal{R}^*$ by D is same as the real view of $\mathcal{R}^*$, and the probability that D outputs 1 given random tuples $(g_1, \ldots, g_6)$ and $(g_1^+, \ldots, g_6^+)$ is equal to the success probability of $\mathcal{R}^*$ to guess the committed values in the definition of hiding property. This must be nonnegligibly larger than $1/2$ by the contradictive assumption.

Hence, D has non-negligible advantage to distinguish between a pair of linear tuples and a pair of random tuples. Such D, by standard argument, implies a distinguisher between a linear tuple and a random tuple with non-negligible advantage, contradicting to the decisional linear assumption. □

Second, we show the determining property (Definition 2) of the proposed scheme.

Proposition 2. *Under the discrete logarithm assumption and the knowledge of exponent assumption on G, the proposed commitment scheme is determining in the global CRS-KR model.*

Proof. Since homomorphic commitment with non-linear tuples is perfectly binding, it is enough to show that any feasible adversarial sender $\mathcal{S}^*$ can generate linear one-time commitment keys $(g_1, \ldots, g_6)$ in (both of) the two parallel executions of the subscheme only with negligible probability.

Suppose a feasible adversary $\mathcal{S}^*$ generates linear tuples $(g_1, \ldots, g_6), (g_1^+, \ldots, g_6^+)$ as its one-time commitment keys in either of the two parallel executions of the subscheme Σ' for a honest receiver with identity ID (using tag tag) with non-negligible probability. Without loss of generality, we assume $\mathcal{S}^*$ generates linear tuples in the second execution with non-negligible probability.

Using $\mathcal{S}^*$, we construct an efficient invertor I that breaks the discrete-logarithm assumption on G with help of some KEA-extractors. Given $g, g_c (= g^{e_c})$, I sets (g, g_c) in the corresponding part of CRS and simulates a receiver of ID for internally invoked $\mathcal{S}^*$. In an execution of the subscheme, I would receive the second sender-message $(g_i), (h_i), (P_i), (Q_i), \ldots$ from $\mathcal{S}^*$. If the messages passes

the specified test by Σ', there should be some c_i and s_i that satisfy $(g_i, h_i) = (g^{c_i}, {G_x}^{c_i})$ and $(P_i, Q_i) = (g^{s_i}, {g_y}^{s_i})$. Then, I can use some KEA-extractors $H^*_{c_i}$ and $H^*_{s_i}$ to extract such c_i and s_i respectively, that provides I with an opening $a_i (= c_i b'_i / b_i), s_i$ of $A_i = g^{a_i} g_c^{s_i}$ in the first sender-message of $\mathcal{S}^*$. (For example, KEA-adversary H_{c_i} on input (g, g_x) reproduces the exact view of $\mathcal{S}^*$ invoked in I using some auxiliary input w_x, and outputs $(g_i (= g^{c_i}), h_i^{1/(H(tag)+d_x)} (= g_x^{c_i}))$.) By rewinding $\mathcal{S}^*$, I can obtain another opening $\hat{a}_i, \hat{s}_i$ of the same $A_i = g^{\hat{a}_i} g_c^{\hat{s}_i}$ from another second sender-message $(\hat{g}_i), (\hat{h}_i), (\hat{P}_i), (\hat{Q}_i), \ldots$. By the contradictive assumption we can suppose both (g_i) and $(\hat{g}_i)$ are linear, and we will see it means that $a_i \neq \hat{a}_i$ with some i. Then I can compute the desired discrete-log e_c of g_c over g, using such $a_i, s_i, \hat{a}_i, \hat{s}_i$ as $e_c = (\hat{a}_i - a_i)(s_i - \hat{s}_i)^{-1}$. More details follow.

Invertor I on input $(g, g_c\ (= g^{e_c}))$:

1. (Simulate a receiver for $\mathcal{S}^*$.) I invokes $\mathcal{S}^*$ and simulates an honest receiver for $\mathcal{S}^*$ as follows, for each of the two parallel executions of the subscheme.
   (a) (First execution.) For the first execution, I honestly generates CRS and public key for $\mathcal{S}^*$ and simulates the receiver in a completely honest way.
   (b) (Second execution.) For the second execution, I simulates the receiver in a following way, using its input (g, g_c) as part of CRS.
      i. (Emulate a public/private-key pair and CRS.) I selects a random δ from ${\mathbb{Z}_q}^*$, sets $g_{ID} = g_c^{\delta}$ and gives g_{ID} as receiver's public-key (for the second execution) to $\mathcal{S}^*$. I also selects random e_x, d_x, e_y from ${\mathbb{Z}_q}^*$ under constraint that $H(tag)+d_x \neq 0$ and sets $g_x = g^{e_x}$, $g_y = g^{e_y}$, $h_x = g_x^{d_x}$ (with g in the input). I gives (g, g_c, g_x, h_x, g_y) as CRS (for the second execution) to $\mathcal{S}^*$, using g, g_c in the input.
      ii. (Emulate the challenge.) Receiving the first sender-message $(A_i), (U_i)$ from $\mathcal{S}^*$, I takes random (b_i) from ${\mathbb{Z}_q}^*$ and sends them to $\mathcal{S}^*$.
      iii. (Extract values generated by $\mathcal{S}^*$.) Receiving the second sender-message $(g_i), \ldots, (b'_i), \ldots$ from $\mathcal{S}^*$, I extracts values $(c_i), (a_i), (s_i)$ as follows, after verifying the equations specified by Σ'.
         A. (Extract (c_i)) I sets $w_x = (\delta, g_c, d_x, e_y, b_1, \ldots, b_6, \text{coins1})$ (where coins1 denotes the coins used to honestly simulate the first execution) and for $i = 1, \ldots, 6$, I invokes KEA-extractor $H^*_{c_i}$ on input $(g, g_x; w_x)$ and obtains the result c_i to compute $a_i = c_i b'_i / b_i$. (Note that e_x is independent of w_x.)
         B. (Extract (s_i)) I sets $w_y = (\delta, g_c, e_x, d_x, b_1, \ldots, b_6, , \text{coins1})$ and for $i = 1, \ldots, 6$, I invokes KEA-extractor $H^*_{s_i}$ on input $(g, g_y; w_y)$ and obtains the result s_i. (Note that e_y is independent of w_y.)
      iv. (Rewind and extract new values.) I rewinds $\mathcal{S}^*$ to Step 1(b)ii and repeats the processes with new challenge $(\hat{b}_i)$ to get the new extracted values $(\hat{c}_i), (\hat{a}_i), (\hat{s}_i)$, receiving another second message $(\hat{g}_i), \ldots, (\hat{b}'_i), \ldots$ from $\mathcal{S}^*$.
2. (Compute the discrete log.) Finally,
   (a) If there exists some i satisfying $b'_i \neq \hat{b}'_i$, then I computes the discrete-log e_c using $w_i, b'_i, \hat{w}_i, \hat{b}'_i$ (as shown below).
   (b) Otherwise if there exists some i satisfying $a_i \neq \hat{a}_i$, then I computes the discrete-log e_c using $a_i, s_i, \hat{a}_i, \hat{s}_i$ (as shown below).
   (c) Otherwise I aborts.

It is direct to see I perfectly simulates a real view of $\mathcal{S}^*$ using its input (g, g_c) as part of CRS of the second execution. Then, I invokes KEA-extractors $H^*_{c_i}$ at Step 1(b)iiiA and $H^*_{s_i}$ at Step 1(b)iiiB, respectively twice, by rewinding $\mathcal{S}^*$. From now on we condition ourselves on the non-negligible event in which $\mathcal{S}^*$ generates linear tuples $(g_1, \ldots, g_6), (g_1^+, \ldots, g_6^+)$ as its one-time commitment keys in both of the two parallel executions, and completes the commitment phase. For a while, assume that

Claim 1 *The extracted values (c_i) (or $(\hat{c}_i)$) at Step 1(b)iiiA and (s_i) (or $(\hat{s}_i)$) at Step 1(b)iiiB satisfy $g_i = g^{c_i},\ P_i = g^{s_i}$ (or $\hat{g}_i = g^{\hat{c}_i},\ \hat{P}_i = g^{\hat{s}_i}$) for $i = 1, \ldots, 6$ with overwhelming probability.*

Under the claim, we can modify the right-hand side of the verifier equation as

$$\begin{aligned} e(A_i, g^{b_i/b'_i}) &= e(g_i, g)\, e(P_i, g_c^{b_i/b'_i}) \\ &= e(g^{c_i}, g)\, e(g^{s_i}, g_c^{b_i/b'_i}) \;=\; e(g, g^{c_i b'_i/b_i} g_c^{s_i})^{b_i/b'_i} \end{aligned}$$

that means $A_i = g^{c_i b'_i/b_i} g_c^{s_i} = g^{a_i} g_c^{s_i}$ for $i = 1, \ldots, 6$. Similarly, we have $A_i = g^{\hat{a}_i} g_c^{\hat{s}_i}$ for $i = 1, \ldots, 6$.

Suppose there exists some i satisfying $b'_i \neq \hat{b}'_i$ at Step 2a. Then, by the verifier equation on w_i, we have $U_i = g^{w_i} g_{ID}^{b'_i} = g^{\hat{w}_i} g_{ID}^{\hat{b}'_i}$, that means $g_{ID} = g^{(\hat{w}_i - w_i)(b'_i - \hat{b}'_i)^{-1}}$. Then, since $g_{ID} = g_c^{\delta}$, I can compute the desired discrete-log e_c as $e_c = (\hat{w}_i - w_i)(b'_i - \hat{b}'_i)^{-1}\delta^{-1}$.

Else if there exists some i satisfying $a_i \neq \hat{a}_i$ at Step 2b, since we have $A_i = g^{a_i} g_c^{s_i} = g^{\hat{a}_i} g_c^{\hat{s}_i}$ as seen above under Claim 1, it holds that $g_c = g^{(\hat{a}_i - a_i)(s_i - \hat{s}_i)^{-1}}$. Hence, I can compute the discrete-log e_c by $e_c = (\hat{a}_i - a_i)(s_i - \hat{s}_i)^{-1}$.

Otherwise, we have $b'_i = \hat{b}'_i$ and $a_i = \hat{a}_i$ for any i, then it must hold $\hat{c}_i/c_i = \hat{b}_i/b_i$ for any i. In this case, the probability that both tuples $(g_i (= g^{c_i}))$ and $(\hat{g}_i (= g^{\hat{c}_i}))$ are simultaneously linear is negligible (since the ratio $\hat{c}_i/c_i$ is uniformly random) and we see that I reaches Step 2c only with negligible probability.

Thus, to complete the proof, all we need is to prove Claim 1.

Proof (of Claim 1). KEA-extractors $H^*_{c_i}$ correspond to the following KEA-adversaries H_{c_i}.

KEA-Adversary H_{c_i} on input $(g, g_x;\ w_x = (\delta, g_c, d_x, e_y, b_1, \ldots, b_6, \text{coins1}))$:

1. (Reproduce the view of $\mathcal{S}^*$.) H_{c_i} invokes $\mathcal{S}^*$ and reproduces the view of $\mathcal{S}^*$ using w_x as follows, for each of the two parallel executions.
   (a) (First execution.) H_{c_i} takes coins1 in w_x and uses it to honestly simulate the first execution of the subscheme.
   (b) (Second execution.) For the second execution, H_{c_i} simulates the receiver in a following way.
      i. (Emulate a public/private-key pair and CRS.) H_{c_i} takes δ and g_c from its auxiliary input w_x to regenerate $g_{ID} = g_c^{\delta}$ and gives it as receiver's public-key (for the second execution) to $\mathcal{S}^*$. H_{c_i} takes d_x, e_y from w_x to regenerate $h_x = g_x^{d_x}, g_y = g^{e_y}$ and gives (g, g_c, g_x, h_x, g_y) as CRS to $\mathcal{S}^*$.
      ii. (Emulate the challenge.) Receiving the first sender-message $(A_i), (U_i)$ from $\mathcal{S}^*$, H_{c_i} takes (b_i) from w_x and sends them to $\mathcal{S}^*$.
      iii. (Generate an output.) Receiving the response $(g_i), (h_i), \ldots$ from $\mathcal{S}^*$, H_{c_i} picks up g_i and h_i from it and outputs $(g_i, h_i^{1/(H(tag)+d_x)})$. (Recall I generated d_x so that $H(tag) + d_x \neq 0$.)

On input (g, g_x), H_{c_i} reproduces the exact view of $\mathcal{S}^*$ invoked in I using the auxiliary input w_x, and generates (g_i, h_i) satisfying $\mathrm{DH}(g, G_x, g_i, h_i)$ with the same probability as $\mathcal{S}^*$ in I does so. When (g, G_x, g_i, h_i) is a DH-tuple and $g_i = g^{c_i}$, we have $h_i^{1/(H(tag)+d_x)} = G_x^{c_i/(H(tag)+d_x)} = g_x^{c_i}$. (Here, recall that I generated d_x so that $H(tag) + d_x \neq 0$.) Thus H_{c_i} outputs $(g^{c_i}, g_x^{c_i})$ on input (g, g_x). As directly verified, the items in w_x are independent of the discrete-log e_x of g_x over g. Hence, by the knowledge of exponent assumption, we see that the corresponding KEA-extractor $H^*_{c_i}$ (in I) outputs c_i satisfying $g_i = g^{c_i}$ only with negligible exception (whenever $\mathcal{S}^*$ generates valid (g_i, h_i)).

KEA-adversary H_{s_i} on input $(g, g_y; w_y)$ proceeds as H_{c_i} and outputs $(P_i = g^{s_i}, Q_i = {g_y}^{s_i})$. By a similar argument, we see $H^*_{s_i}$ in I outputs s_i satisfying $P_i = g^{s_i}$ only with negligible exception. □

□

Now we show the straight-line equivocal-extractability (Definition 4) of the proposed scheme.

Proposition 3. *Under the decisional linear assumption and the knowledge of exponent assumption on G, the proposed commitment scheme is straight-line equivocal-extractable in the global CRS-KR model.*

Proof Idea. To show the proposition, we need to construct a simulator EQV of a left party and an extractor EXT of values committed to by any feasible adversary A for a right party, that satisfy Definition 4.

EQV is given A's secret key e_{ID^*} and proves that it knows e_{ID^*} in the second sender-message, instead of proving its honesty to the generation of one-time commitment key. By doing this, EQV can use a linear tuple as the one-time commitment key for its commitment. Since linear tuples define equivocal commitments, EQV is able to adaptively simulate left parties without knowing values actually being committed to. We show such simulation of left parties by EQV is indistinguishable from real left parties for feasible adversaries A under the decisional linear assumption on G with help of some KEA-extractors. (To show the indistinguishability, we need to extract the values committed to by A, that is enabled by using some KEA-extractors.)

EXT extracts the value committed to by A from its view $view_A$ using the trapdoor information of CRS. For that sake, EXT needs to extract the coins c_i^*, used by A to generate its one-time commitment key. EXT invokes some KEA extractors on input (g, g_x) using (some part of) the view $view_A$ as auxiliary information to extract such coins c_i^*. Here, we need a care to use the knowledge of exponent assumption in a right way for such KEA-extractors, because the view $view_A$ itself is dependent on the discrete-log between g and g_x that are included in CRS. (We can use the knowledge of exponent assumption only for KEA-extractors that uses auxiliary information that is independent of its input.) We will carefully examine distribution of $view_A$ and pick up some portion from $view_A$ that are independent of that discrete-log, making use of zero-knowledge simulators for proofs P_i.

In the course of the proof, one key point is to ensure that the adversary A cannot use linear tuples as its own one-time commitment key, even A receives linear tuples as one-time commitment keys from the simulated left party by EQV. This impossibility is brought by the use of tag tag in the scheme for generation of $G_x = g_x^{H(tag)} h_x$, that is in turn used for the generation of (h_i) in the second sender-message. This dependency on tag prevents the flipped coins generated for one-time commitment keys from being copied from/to other sessions. The detailed proof is in Section A.

By Theorem 1, Proposition 2 and Proposition 3, we have

Theorem 2. *Under the decisional linear assumption and the knowledge of exponent assumption on G, the proposed commitment scheme is adaptive-deniable-concurrent non-malleable in the global CRS-KR model.*

4 Conclusion

We defined a notion of adaptive-deniable-concurrent non-malleable commitments, that captures the composability, deniability and adaptivity at once for commitment schemes. Then we defined a more-easy-to-prove property of straight-line equivocal-extractability of commitment schemes and proved that it yields the adaptive-deniable-concurrent non-malleability in the global CRS-KR model. We also gave a construction of straight-line equivocal-extractable (especially, adaptive-deniable-concurrent non-malleable) commitment scheme in the global CRS-KR model, under the decisional linear assumption and the knowledge of exponent assumption on bilinear groups. The scheme is practically efficient, using a constant number of pairing computations and three-round exchanges of linear-size messages.

References

1. M. Abe and S. Fehr, Perfect NIZK with Adaptive Soundness, pp. 118-136, Proc. of TCC, LNCS 4392, 2007.
2. B. Barak, How to go beyond the black-box simulation barrier, Proc. 42nd FOCS, pp. 106-115, IEEE, 2001.
3. R. Canetti, Universally Composable Security: A New Paradigm for Cryptographic Protocols, pp. 136-145, Proc. of FOCS, 2001.
4. R. Canetti, Y. Dodis, R. Pass, S. Walfish, Universally Composable Security with Global Setup, pp. 61-85, Proc. of TCC 2007, LNCS 4392.
5. R. Canetti, M. Fischlin, Universally Composable Commitments, pp. 19-40, Proc. of Crypto, 2001.
6. R. Cramer, I. Damgård and B. Schoenmakers, Proofs of Partial Knowledge and Simplified Design of Witness Hiding Protocols, pp. 174-187, Proc. of Crypto 1994, LNCS 839.
7. R.Canetti, S.Halevi and J.Katz, Chosen-ciphertext security from identity-based encryption, EUROCRYPT 2004, LNCS 3027, pp.207-222, 2004.
8. G. D. Crescenzo, Y. Ishai and R. Ostrovsky, Non-interactive and Non-malleable Commitment, pp. 141-150, STOC 1998.
9. G. D. Crescenzo, J. Katz, R. Ostrovsky and A. Smith, Efficient and Non-interactive Non-malleable Commitments, pp. 40-59, Proc. of EuroCrypt 2001, LNCS 2045.
10. I. Damgård, Towards practical public-key cryptosystems provably-secure against chosen-ciphertext attacks, pp. 445-456, Proc. of CRYPTO '91, LNCS 576.
11. I. Damgård, Efficient Concurrent Zero-Knowledge in the Auxiliary String Model, pp. 418-430, Proc. of EUROCRYPT 2000, LNCS 1807.
12. I. Damgård and J. Groth, Non-interactive and reusable non-malleable commitment schemes, pp. 426-437, Proc. of STOC 2003, ACM Press.
13. I. Damgård and J. B. Nielsen, Perfect Hiding and Perfect Binding Universally Composable Commitment Schemes with Constant Expansion Factor, pp. 581-596, Proc. of Crypto 2002, LNCS 2442.
14. D. Dolev, C. Dwork and M. Naor, Non-Malleable Cryptography, SIAM Journal on Computing, Vol. 30(2), pp. 391–437, 2000.
15. M. Fischlin and R. Fischlin, Efficient Non-malleable Commitment Schemes, pp. 413-431, Proc. of Crypto 2000, LNCS 1880.
16. J. Groth, R. Ostrovsky, and A. Sahai, Perfect non-interactive zero knowledge for NP, 339-358, EUROCRYPT, LNCS 4004, 2006.
17. H. Lin, R. Pass, M. Venkitasubraamaniam, Concurrent Non-malleable Commitments from Any One-way Function, TCC 2008, LNCS 4948, pp. 571–588, 2008.
18. P. D. MacKenzie, M. K. Reiter and K. Yang, Alternatives to Non-malleability: Definitions, Constructions, and Applications, pp. 171-190, Proc. of TCC 2004, 2004.

19. R. Ostrovsky, G. Persiano, and I. Visconti, Simulation-Based Concurrent Non-malleable Commitments and Decommitments, TCC 2009, LNCS 5444, pp. 91-108, 2009.
20. R. Pass, On Deniabililty in the Common Reference String and Random Oracle Model, pp. 316-337, Proc. of CRYPTO 2003, LNCS 2729.
21. R. Pass, A. Rosen, Concurrent non-malleable commitments, 563 - 572, FOCS 2005.

A Proof of Proposition 3

Let $\Sigma = (\Sigma'|\Sigma')$ with $\Sigma' = (K, S, R)$ denote the proposed commitment scheme on G. We want to show there exists a feasible algorithm EQV such that for any feasible adversary A, there exists a feasible algorithm EXT and

$$\mathsf{fakeCom}^{\Sigma}_{\mathsf{EQV},A,\mathsf{EXT}}(m, z) \equiv_c \mathsf{realCom}^{\Sigma}_{A}(m, z) \tag{6}$$

with any value m ($\in \mathbb{Z}_q$) and any string z (Definition 4).

For that sake, we define four experiments $\mathbf{Exp}_0, \ldots, \mathbf{Exp}_3$ in the sequel and proves indistinguishability of their outputs step by step. $\mathbf{Exp}_0$ is nothing but the experiment $\mathsf{realCom}^{\Sigma}_{A}(m, z)$ instantiated with our Σ, and the final experiment $\mathbf{Exp}_3$ gives EQV and EXT, proving Equation (6).

$\mathbf{Exp}_0$. Given m and z as input, $\mathbf{Exp}_0$ invokes an adversary A on z. First, A selects a tag tag for the left interaction, its own id ID^* and right-party's id ID. A receives CRS σ, its public/private-key pair (g_{ID^*}, e_{ID^*}) and the right-party's public key g_{ID}, for each of the two parallel executions of the subscheme. Then, $\mathbf{Exp}_0$ simulates the left party that commits to m and the right party for A, honestly. $\mathbf{Exp}_0$ returns trapdoor (τ, τ^+) of the CRSs of both executions, the simulated A's view $view_A$, the determining value m^* of A's commitment in $view_A$ and left-party's decommitment $((m, r, t), (m, r^+, r^+))$ that corresponds to the left-party's commitment in $view_A$. (A full description of $\mathbf{Exp}_0$ is in Section B.)

$\mathbf{Exp}_1$. Only the behavior of the left party is changed as follows. New left party proves the knowledge of A's secret key e_{ID^*}, instead of proving the honesty of the generation of the one-time commitment key. More precisely, for each of the two parallel executions of the subscheme, the new left party computes U_i as $U_i = g^{k_i}$ with random k_i (instead of $U_i = g^{w_i} {g_{ID^*}}^{b'_i}$ with b'_i selected in advance). Moreover, the left party selects a random one-time commitment key ($g_i = g^{c_i}$) independently of $A_i = g^{a_i} g_c^{s_i}$ in the first message. Although proofs $P_i = g^{s_i}$, $Q_i = {g_y}^{s_i}$ are honestly computed using s_i, the response w_i is computed as $w_i = k_i - b'_i e_{ID^*}$ using A's secret key e_{ID^*} for challenge b'_i, that is now computed from A's challenge b^*_i through $b''_i = c_i/a_i$, $b'_i = b^*_i/b''_i$. (A full description of $\mathbf{Exp}_1$ is in Section B.)

Claim 2 *The output of* $\mathbf{Exp}_1$ *is perfectly indistinguishable from the output of* $\mathbf{Exp}_0$.

Proof. The only difference between $\mathbf{Exp}_0$ and $\mathbf{Exp}_1$ is in the order of generation of challenges b'_i and b''_i: Generate random b'_i and set $b''_i = b^*/b'_i$, or generate random $b''_i = c_i/a_i$ and set $b'_i = b^*/b''_i$. This difference does not affect the resulting distribution of left party's messages and the output of $\mathbf{Exp}_1$ distributes exactly as the output of $\mathbf{Exp}_0$. □

$\mathbf{Exp}_2$. Third experiment $\mathbf{Exp}_2$ proceeds just as $\mathbf{Exp}_1$, except that (c_i) is now selected so that (g^{c_i}) becomes a linear tuple, instead of a random tuple. (A full description of $\mathbf{Exp}_2$ is in Section B.)

In order to prove indistinguishability between the outputs of $\mathbf{Exp}_2$ and $\mathbf{Exp}_1$, first we prepare a following lemma.

Lemma 1. *Under the same assumption as Proposition 3, any feasible adversary A outputs a linear tuple $(g_1^*, \ldots, g_6^*)$ as the one-time commitment key for the right interaction in either of the two execution of the subscheme only with negligible probability in $\mathbf{Exp}_1$ (or $\mathbf{Exp}_2$).*

Proof. The proof is similar to the proof of Proposition 2, where we have shown that any feasible sender $\mathcal{S}^*$ in a stand-alone setting can generate a linear tuple $(g_1, \ldots, g_6)$ as the one-time commitment key only with negligible probability, by exhibiting an invertor I that computes the discrete-log of g_c over g with help of the KEA-extractors H_{c_i} and H_{s_i} using such $\mathcal{S}^*$.

To prove this lemma, we again construct an invertor I that computes the discrete-log of g_c over g using such A. In addition to the work done by I in the proof of Proposition 2, the new I needs to simulate the honest left party. Since also this I, that sets up public keys for simulated A, knows A's secret key e_{ID^*}, there is no difficulty for this I to simulate the left party in $\mathbf{Exp}_1$ (or $\mathbf{Exp}_2$).

One subtle point is in the construction of new KEA-adversary H_{c_i}. Recall that the work of H_{c_i} is to compute $(g_i(= g^{c_i}), g_x^{c_i})$ on input (g, g_x) (and $w_x = (\delta, g_c, d_x, e_y, b_1, \ldots, b_6, \text{coins1})$). In the proof of Proposition 2, the trapdoor d_x (in w_x), that is generated to satisfy $H(tag) + d_x \neq 0$, is used to compute the $g_x^{c_i}$ by H_{c_i}. H_{c_i} picked up $h_i\,(= G_x^{c_i})$ from the $\mathcal{S}^*$'s second message and computed $g_x^{c_i} = h_i^{1/(H(tag)+d_x)}$. (Recall $G_x = {g_x}^{H(tag)+d_x}$.) However in the case of this lemma, H_{c_i} must deal with the right interaction and need to compute $g_x^{c_i^*}$ from $h_i\,(= G_x^{*c_i^*})$ in the right interaction with $G_x^* = g_x^{H(tag^*)} h_x$. Now tag^*, that also belongs to the right interaction, is selected by A after it sees CRS (that includes g, g_x). So, we have no guarantee of being $H(tag^*) + d_x \neq 0$ and new H_{c_i} cannot mimic the old H_{c_i} just doing as $g_x^{c_i^*} = h_i^{1/(H(tag^*)+d_x)}$.

To remedy this situation, the new I generates g_x and h_x in CRS in a following manner. First, I selects a random e_x, η from $\mathbb{Z}_q^*$, and sets $g_x = g^{e_x},\ h_x = g_x^{-H(tag)} g^{\eta}$. (Note $G_x = g^{\eta}$ and remember the left tag tag is selected by A before it sees CRS.) Then, H_{c_i} can compute $g_x^{c_i^*}$ as $\left(\frac{h_i^*}{g_i^{*\eta}}\right)^{\frac{1}{H(tag^*)-H(tag)}}$, using η that will be newly added in the auxiliary input w_x. (Note when (g, G_x^*, g_i^*, h_i^*) is a DH-tuple and $g_i^* = g^{c_i^*}$, we have $h_i^* = (G_x^*)^{c_i^*} = (g^{\eta} g_x^{H(tag^*)-H(tag)})^{c_i^*} = g_i^{*\eta} (g_x^{c_i^*})^{H(tag^*)-H(tag)}$. So, $\left(\frac{h_i^*}{g_i^{*\eta}}\right)^{\frac{1}{H(tag^*)-H(tag)}} = g_x^{c_i^*}$.) Because A must choose distinct tag^* from tag (to gain nontrivial advantage), it is guaranteed that $H(tag^*) - H(tag) \neq 0$. □

Claim 3 *Under the same assumption as Proposition 3, the output of $\mathbf{Exp}_2$ is computationally indistinguishable from the output of $\mathbf{Exp}_1$.*

The proof is in the same line to the proof of Proposition 1, where in order to show the hiding property a distinguisher between linear and random tuples was constructed and it simulated an honest sender for adversarial receiver $\mathcal{R}^*$ to gain its advantage. In the proof of this claim, a new distinguisher does the same simulation for the left interaction for adversary A, and in addition it simulates an honest right party and extracts the value committed to by A for the right party in some efficient way, using some KEA-extractors, that is needed for the assumed distinguisher to distinguish between the two experiments.

Proof. Suppose that the output of $\mathbf{Exp}_2$ is distinguishable from the output of $\mathbf{Exp}_1$ by some efficient distinguisher D_{12} with some non-negligible advantage (with respect to m and z).

Using such distinguisher D_{12}, we construct an efficient (non-uniform) distinguisher D_{LIN} between a *pair* of linear tuples and a *pair* of random tuples with non-negligible advantage, that would contradict to the decisional linear assumption on G.

Given a pair of tuples $(g_1, \ldots, g_6), (g_1^+, \ldots, g_6^+)$ as inputs, D_{LIN} simulates the left party for A in such a way that the $(g_1, \ldots, g_6), (g_1^+, \ldots, g_6^+)$ would occupy one-time commitment keys of the left interactions in the two parallel executions. Note that, due to the modification introduced in $\mathbf{Exp}_1$, one-time commitment keys included in the second sender-message are being chosen independently from the first sender-message $(A_i), (U_i)$. So, such simulation is quite easy. After generating the view of A, D_{LIN} invokes KEA-extractors $H^*_{c_i}$, that are essentially same as the ones used by I in the proof of Lemma 1, to extract coins c_i^* satisfying $g_i^* = g^{c_i^*}$ for the one-time commitment key (g_i^*) in the right interaction. Because by Lemma 1 the commitment keys $(g_1^*, \ldots, g_6^*)$ can be supposed to be non-linear, D_{LIN} is able to compute g^{m^*} (with the determining value m^* of A's commitment in the right interactions) as $g^{m^*} = (M_1^{*1/c_4^*} M_2^{*1/c_5^*} M_3^{*-1/c_6^*})^{1/(c_1^*/c_4^* + c_2^*/c_5^* - c_3^*/c_6^*)}$ using those c_i^*, and as well as g^{+m^*}. This means that D_{LIN} can extract m^* by using another KEA-extractor (that corresponds to another KEA-adversary that outputs (g^{m^*}, g^{+m^*}) on input (g, g^+)) and then can invoke the assumed distinguisher D_{12} on m^* (and other easy-to-collect items) to distinguish the given tuples. A formal description of D_{LIN} is given in Figure 5.

- **Distinguisher** D_{LIN} on input $((g_1, \ldots, g_6), (g_1^+, \ldots, g_6^+);\ (z, m))$:
  1. (Simulate the left and right parties for A.) D_{LIN} invokes A on z. Receiving ID^*, ID and tag from A, D_{LIN} simulates the left and right parties for A as follows, for each of the two parallel executions of the subscheme.
     (a) (Emulate a public/private-key pair and CRS.) D_{LIN} selects random e_{ID^*} from ${\mathbb{Z}_q}^*$, a random g from G and sets $g_{ID^*} = g^{e_{ID^*}}$. D_{LIN} also selects a random g_{ID} from G. D_{LIN} gives to A (g_{ID^*}, e_{ID^*}) as A's public/private-key pair and g_{ID} as the right-party's public key. Further, D_{LIN} selects random e_c, e_x, e_y, η from ${\mathbb{Z}_q}^*$ and sets $g_c = g^{e_c},\ g_x = g^{e_x},\ g_y = g^{e_y},\ h_x = g_x^{-H(tag)} g^{\eta}$. (Note that $G_x = g^{\eta}$.) D_{LIN} gives (g, g_c, g_x, h_x, g_y) as CRS to A.
     (b) (Emulate a first left-party message.) Receiving a request from A, D_{LIN} chooses random $(s_i), (k_i), (b_i'')$ from $\mathbb{Z}_q^*$ and computes, for $i = 1, \ldots, 6$, $A_i = {g_i}^{1/b_i''} g_c^{s_i},\ U_i = g^{k_i}$. (Note when $g_i = g^{c_i}$ and $c_i = a_i b_i''$, we have ${g_i}^{1/b_i''} = g^{a_i}$ as in $\mathbf{Exp}_1$ (or $\mathbf{Exp}_2$).) D_{LIN} sends $(A_i), (U_i)$ to A.
     (c) (Emulate a first right-party message.) Receiving a first message $tag^*, (A_i^*), (U_i^*)$ to the right party from A, D_{LIN} chooses random (b_i) from ${\mathbb{Z}_q}^*$ and sends them to A.
     (d) (Emulate a second left-party message.) Receiving a first message (b_i^*) to the left party from A, D_{LIN} computes, for $i = 1, \ldots, 6$, $h_i = {g_i}^{\eta},\ P_i = g^{s_i},\ Q_i = {g_y}^{s_i}$. (Note when $g_i = g^{c_i}$, we have $h_i = g_i^{\eta} = (g^{\eta})^{c_i} = G_x^{c_i}$ as in $\mathbf{Exp}_1$ (or $\mathbf{Exp}_2$).) In addition, D_{LIN} computes $w_i = k_i - b_i' e_{ID^*}$ for $b_i' = b_i^*/b_i''$ for $i = 1, \ldots, 6$, and computes homomorphic commitment M_1, M_2, M_3 to m using its input (g_i) as commitment key. D_{LIN} sends the second sender-message $(g_i), (h_i), \ldots, (M_i)$ to A.
     (e) (Complete commitment phases.) Receiving a second message $(g_i^*), (h_i^*), \ldots, (M_i^*)$ to the right party from A, D_{LIN} checks the verifier equation as specified by Σ'. If any of them is not true, D_{LIN} aborts with a random bit. Else let $w = ((g_1, \ldots, g_6), \mathsf{coins})$, where coins denotes the coins used to simulate this execution of subscheme so far by D_{LIN}, except the coins used for generating the first element g in the CRS.
  2. (Generate output.) Let g and g^+ be the first elements of CRSs in the two executions of the subscheme. D_{LIN} invokes KEA-extractor H^* on $(g, g^+;\ z, w, w^+)$ to get m^*. Then, D_{LIN} invokes the assumed distinguisher D_{12} on $(\{\tau, \tau^+\}, view_A, m^*, \{(m, r, t), (m, r^+, t^+)\})$ and outputs its output.

Fig. 5. Distinguisher D_{LIN}.

As easily seen (from comments in the description), D_{LIN} perfectly simulates the view of A in $\mathbf{Exp}_1$ if the input $(g_i), (g_i^+)$ are random, and perfectly simulates the view of A in $\mathbf{Exp}_2$ if the input $(g_i), (g_i^+)$ are linear. Then, D_{LIN} extracts the value m^*, that is supposed to be committed to by A for the right party, by using KEA-extractor H^*. Assuming the extracted m^* is in fact the one committed to by A, D_{LIN} is now able to use the assumed distinguisher D_{12} on m^* and the others, to distinguish the input tuples are linear or random, contradicting to the assumption.

Thus, now all we need to show is that the KEA-extractor H^*, that corresponds to following KEA-adversary H, extracts the right determining value m^* of A's commitment in the right interaction only with negligible exceptions.

KEA-Adversary H on input $(g, g^+;\ z, m, w, w^+)$:

1. ($\mathsf{Reproduce\ the\ view\ of}\ A$.) H invokes A on z. Receiving ID^*, ID, tag from A, H reproduces the view of A using the coins given in w, w^+, for each of the two parallel executions of the subscheme.
2. ($\mathsf{Extract\ coins\ generated\ by}\ A$.) For each of the above two parallel executions, H extracts the coins (c_i^*) that defines the one-time commitment key (g_i^*) in the right interactions. Namely, H sets $w_x = (z, e_{ID^*}, g_{ID}, g_c, \eta, e_y, g_1, \ldots, g_6, \mathsf{coin}_L, b_1, \ldots, b_6, w^+)$ (where notation is the same as in D_{LIN} besides coin_L, that denotes the coins used to simulate the left party in this execution) and invokes KEA-extractor $H^*_{c_i}$ on input $(g, g_x; w_x)$ to obtain c_i^* satisfying $g_i^* = g^{c_i^*}$ for $i = 1, \ldots, 6$. Then, H computes $u = (M_1^{*1/c_4^*} M_2^{*1/c_5^*} M_3^{*-1/c_6^*})^{1/(c_1^*/c_4^* + c_2^*/c_5^* - c_3^*/c_6^*)}$. (Note it should be $u = g^{m^*}$ with determining value m^* of A's homomorphic commitment in this execution.)
3. ($\mathsf{Generate\ output}$.) Output a pair of extracted values (u, u^+) of the two executions.

Given (g, g^+) as input, KEA-adversary H uses the coins given in the auxiliary input w, w^+ to reproduce the exact view of A in D_{LIN} and then extracts coins c_i^* generated by A, that are expected to satisfy $g_i^* = g^{c_i^*}$ by using KEA-extractors $H^*_{c_i}$ on (g, g_x) and $w_x = (z, e_{ID^*}, g_{ID}, g_c, \eta, e_y, g_1, \ldots, g_6, \mathsf{coin}_L, b_1, \ldots, b_6, u$ For a while, assume the coins c_i^* are right. Then, since $(g_1^* = g^{c_1^*}, \ldots, g_6^* = g^{c_6^*})$ is not linear by Lemma 1 (only with negligible exceptions), it defines perfectly binding homomorphic commitment and so H can compute the value $u = (M_1^{*1/c_4^*} M_2^{*1/c_5^*} M_3^{*-1/c_6^*})^{1/(c_1^*/c_4^* + c_2^*/c_5^* - c_3^*/c_6^*)}$ that must be equal to g^{m^*} with $m^* = \det(M_1^*, M_2^*, M_3^*)$. The output of H is a pair (u, u^+) of such u for the two executions of the subscheme. Thus, H outputs (g^{m^*}, g^{+m^*}) on input (g, g^+), and since clearly $\mathrm{Log}_g(g^+)$ is independent from the auxiliary input (z, m, w, w^+), we see the corresponding H^* extracts the right determining value m^* of A's commitment only with negligible exceptions, by the knowledge of exponent assumption, as desired.

Now, all we have to do is to show the above KEA-extractors $H^*_{c_i}$, that correspond to the following KEA-adversaries, extract the right coins c_i^* satisfying $g_i^* = g^{c_i^*}$ only with negligible exceptions. As in the proof of Lemma 1, given (g, g_x) as input, H_{c_i} reproduces the exact view of A invoked in H using the auxiliary input w_x, and picks up g_i^* and h_i^* from the view and outputs $\left(g_i^* (= g^{c_i^*}), \left(\frac{h_i^*}{g_i^{*\eta}} \right)^{\frac{1}{H(tag^*) - H(tag)}} (= g_x^{c_i^*}) \right)$. Namely, H_{c_i} outputs $(g^{c_i^*}, g_x^{c_i^*})$ on input (g, g_x). In addition, as directly verified, all items in w_x are independent of the discrete-log e_x of g_x over g. Hence, by the knowledge of exponent assumption, $H^*_{c_i}$ in H must output c_i^* satisfying $g_i^* = g^{c_i^*}$ only with negligible exception. This completes the proof of Claim 3. □

$\mathbf{Exp}_3$. In the final experiment $\mathbf{Exp}_3$, the simulated left party is changed to commit to a dummy value m_0 through homomorphic commitment $M_1 = g_1^{m_0} g_4^{r_0},\ M_2 = g_2^{m_0} g_5^{t_0},\ M_3 = g_3^{m_0} g_6^{r_0 + t_0}$, instead

of committing to the true value m, for each of the two executions of subscheme. Since by the modification introduced in $\mathbf{Exp}_2$ the one-time commitment key $(g_i(=g^{c_i}))$ of the left interaction is linear, its decommitment to the true value m (to be included in the output) exists and is computed by $r = r_0 + \frac{c_1}{c_4}(m_0 - m),\ t = t_0 + \frac{c_2}{c_5}(m_0 - m)$. In addition, in $\mathbf{Exp}_3$, the computation of the determining value $m^* = \det(M_1^*, M_2^*, M_3^*)$ of A's commitment is changed. It is now efficiently extracted by using KEA-extractor H^* on $(g, g^+;\ z, m, w, w^+)$ with $w = (e_{ID^*}, g_{ID}, \tau, (A_i), (U_i), (b_i), (g_i), (b_i'), (M_i))$. (A full description of $\mathbf{Exp}_3$ is in Section B.)

Claim 4 *The output of* $\mathbf{Exp}_3$ *is statistically indistinguishable from the output of* $\mathbf{Exp}_2$.

Proof. In $\mathbf{Exp}_2$, the homomorphic commitment M_1, M_2, M_3 in the left interaction is perfectly hiding and equivocal since the used one-time commitment key $g_1, \ldots, g_6$ is linear. So, it is trivial to see the $view_A$ in $\mathbf{Exp}_3$ is exactly distributed as the $view_A$ in $\mathbf{Exp}_2$.

Then, all we need to show is that given input $(g, g^+;\ z, m, w, w^+)$, H^* in $\mathbf{Exp}_3$ correctly outputs the determining value $m^* = \det(M_1^*, M_2^*, M_3^*)$ of A's commitment in the right interaction only with negligible exceptions. This H^* is similar as H^* used by D_{LIN} in the proof of Claim 3. However, there is an important difference. As seen below, the new H (that corresponds to the new H^*) also uses extractors $H_{c_i}^*$ of c_i^* on input (g, g_x). However, in the current setting, the information given to this $H_{c_i}^*$ through auxiliary input w_x will contain (part of) A's view $(A_i), (U_i), (b_i), (g_i), (b_i'), (M_i)$, instead of the used coins as before. So, as to the new H_{c_i}, there is no direct guarantee that the discrete-log e_x of the given input (g_x, g) is independent of the given auxiliary input w_x. (The left party's second message, especially $(b_i' = b_i^*/b_i'')$, depends on the A's challenge (b_i^*), that possibly depends on the CRS, that includes g, g_x.) So, before describing the new KEA-adversary H, we need to give an equivalent code for simulating the left party in $\mathbf{Exp}_3$, to avoid such problem.

Fix arbitrary left party's first message $(A_i), (U_i)$ and arbitrary right party's first message (b_i) in $\mathbf{Exp}_3$. (Note then A's challenge (b_i^*) for the left party is also fixed.) Then, in $\mathbf{Exp}_3$, the second left-party message $(g_i), (h_i), \ldots, (w_i)$ (excluding (M_i) which is clearly independent of g, g_x) is being generated by the following procedure. (Note the fixed $(U_i = g^{k_i})$ implicitly determines k_i.)

- The original procedure:
  1. Take a random tuple $(c_1, \ldots, c_6)$ so that $(g_1 = g^{c_1}, \ldots, g_6 = g^{c_6})$ is linear and a random tuple $(s_1, \ldots, s_6)$ from $\mathbb{Z}_q^*$. (Note s_i implicitly determines a_i through the fixed $A_i = g^{a_i} g_c^{s_i}$.)
  2. For $i = 1$ to 6, compute $g_i = g^{c_i},\ h_i = G_x^{c_i},\ P_i = g^{s_i},\ Q_i = g_y^{s_i},\ b_i'' = c_i/a_i,\ b_i' = b_i^*/b_i'',\ w_i = k_i - b_i' e_{ID^*}$.

This procedure is re-written as follows, using trapdoor information τ and the zero-knowledge simulator of proof P_i.

- The second procedure:
  1. Take a linear tuple $(g_1, \ldots, g_6)$ from G and a random tuple $(b_1'', \ldots, b_6'')$ from $\mathbb{Z}_q$.
  2. For $i = 1$ to 6, compute $h_i = g_i^{\eta},\ P_i = (A_i g_i^{-1/b_i''})^{1/e_c},\ Q_i = P_i^{e_y},\ b_i' = b_i^*/b_i'',\ w_i = k_i - b_i' e_{ID^*}$, with $\eta = e_x(H(tag) + d_x)$.

We can select random (b_i'), instead of random (b_i''), preserving the distribution and the above procedure is re-written as follows.

- The third procedure:

1. Take a linear tuple $(g_1, \ldots, g_6)$ from G and a random tuple $(b'_1, \ldots, b'_6)$ from $\mathbb{Z}_q$.
2. For $i = 1$ to 6, compute $h_i = g_i^{\eta},\ b''_i = b^*_i / b'_i,\ P_i = (A_i g_i^{-1/b''_i})^{1/e_c},\ Q_i = P_i^{e_y},\ w_i = k_i - b'_i e_{ID^*}$.

The third procedure shows that the second left-party message includes the three independent random components $(g_i), (b'_i), (M_i)$ and the other items $(h_i), (P_i), (Q_i), (w_i)$ are determined by them. Thus, we see that the discrete-log e_x of g_x over g is independent of the part of $view_A$ consisting of $(A_i), (U_i), (b_i)$ and $(g_i), (b'_i), (M_i)$.

Now we describe the KEA-adversary H that uses the above third procedure to simulate the left-party messages.

KEA-Adversary H on input $(g, g^+;\ z, w = (e_{ID^*}, g_{ID}, \tau, (A_i), (U_i), (b_i), (g_i), (b'_i), (M_i)), w^+)$:

1. (Reproduce the view of A.) H invokes A on z. Receiving ID^*, ID, tag from A, H reproduces the view of A using w as follows, for each of the two parallel executions of the subscheme.
   (a) (Emulate a public/private-key pair and CRS.) H takes trapdoor information τ from w to regenerate A's public/private-key pair (g_{ID^*}, e_{ID^*}), right-party's public key g_{ID} and CRS (g, g_c, g_x, h_x, g_y), and gives them to A. H records $\eta = e_x(H(tag) + d_x)$ for later use.
   (b) (Emulate a first left-party message.) Receiving a request for a first left-party message from A, H takes $(A_i), (U_i)$ from w and sends them to A.
   (c) (Emulate a first right-party message.) Receiving a first message $tag^*, (A^*_i), (U^*_i)$ to the right-party from A, H takes (b_i) from w and sends them to A.
   (d) (Emulate a second left-party message.) Receiving a first message (b^*_i) to the left-party from A, H takes $(g_i), (b'_i), (M_i)$ from w, computes $h_i = g_i^{\eta},\ b''_i = b^*_i / b'_i,\ P_i = (A_i g_i^{-1/b''_i})^{1/e_c},\ Q_i = P_i^{e_y}, w_i = k_i - b'_i e_{ID^*}$, and sends $(g_i), (h_i), (P_i), (Q_i), (b'_i), (w_i), (M_i)$ to A.
   (e) (Extract coins generated by A.) Receiving the second message $(g^*_i), (h^*_i), \ldots, (M^*_i)$ to the right-party from A, H extracts the coins (c^*_i) that A generated. Namely, H sets $w_x = (z, e_{ID^*}, g_{ID}, e_c, \eta, e_y, (A_i), (U_i), (b_i), (g_i), (b'_i), (M_i), view^+_A)$ (where $view^+_A$ denotes the view of A in the another execution of the subscheme) and invokes KEA-extractor $H^*_{c_i}$ on input $(g, g_x; w_x)$ to obtain c^*_i, for $i = 1, \ldots, 6$.
   (f) (Extract g^{m^*} generated by A.) H computes $u = (M_1^{*1/c^*_4} M_2^{*1/c^*_5} M_3^{*-1/c^*_6})^{1/(c^*_1/c^*_4 + c^*_2/c^*_5 - c^*_3/c^*_6)}$.
2. (Generate output.) Return a pair of the extracted values (u, u^+) at Step 1f.

Given (g, g^+) as input, KEA-adversary H reproduces the exact view of A in $\mathbf{Exp}_3$ by using the third procedure on those independent random components given in the auxiliary input w, and then extracts A's coins c^*_i that are expected to satisfy $g^*_i = g^{c^*_i}$ by KEA-extractor $H^*_{c_i}$ on (g, g_x) and $w_x = (z, e_{ID^*}, g_{ID}, e_c, \eta, e_y, (A_i), (U_i), (b_i), (g_i), (b'_i), (M_i), view^+_A)$. For a while, assume the expectations are right. Then, since $(g^*_1 = g^{c^*_1}, \ldots, g^*_6 = g^{c^*_6})$ is not linear by Lemma 1, it defines perfectly binding homomorphic commitment and so H can compute the value $u = (M_1^{*1/c^*_4} M_2^{*1/c^*_5} M_3^{*-1/c^*_6})^{1/(c^*_1/c^*_4 + c^*_2/c^*_5 - c^*_3/c^*_6)}$ that must be equal to g^{m^*} with $m^* = \det(M^*_1, M^*_2, M^*_3)$. The output of H is a pair (u, u^+) of such u for the two executions of the subscheme. Thus, H outputs (g^{m^*}, g^{+m^*}) on (g, g^+). Since clearly the discrete-log of g^+ over g is independent from the auxiliary input (z, m, w, w^+), by the knowledge of exponent assumption, the corresponding KEA-extractor H^* extracts the right determining value m^* of A's commitment only with negligible exceptions, as desired.

Now, all we have to do is to show that the KEA-extractor $H^*_{c_i}$, that corresponds to the following KEA-adversary H_{c_i}, extracts the coins c_i^* satisfying $g_i^* = g^{c_i^*}$ only with negligible exceptions. Given (g, g_x) as input, H_{c_i} reproduces the exact view of A invoked in H using the auxiliary input w_x by using the third procedure, and picks up g_i^* and h_i^* from them and outputs $\left(g_i^*(= g^{c_i^*}), (h_i^*/g_i^{*\eta})^{1/(H(tag^*)-H(tag))}(= g_x^{c_i^*})\right)$. As discussed before, the items in w_x (especially the ones that belongs to the view of A) was chosen to be independent of the discrete log e_x of g_x with g. Hence, by the knowledge of exponent assumption, $H^*_{c_i}$ outputs c_i^* satisfying $g_i^* = g^{c_i^*}$ only with negligible exceptions. That completes the proof of Claim 4. □

By Claims 2, 3 and 4, Equation (6) is satisfied by EQV and EXT in Figure 6, that are immediate from the description of $\mathbf{Exp}_3$ and the third procedure, completing the proof of Proposition 3. □

- **Simulator** EQV on input (tag, e_{ID^*}, m_0) does as follows, for each of the two parallel executions of the subscheme:
  1. (Emulate a first left-party message.) Choose random a_i, s_i, k_i from $\mathbb{Z}_q^*$ and compute $A_i = g^{a_i} g_c^{s_i}$, $U_i = g^{k_i}$ for $i = 1, \ldots, 6$. Send $A_1, \ldots, A_6, U_1, \ldots, U_6$.
  2. (Emulate a second left-party message.) Receiving a first message (b_i^*), select random c_i from $\mathbb{Z}_q^*$ so that (g^{c_i}) becomes a linear tuple and compute $g_i = g^{c_i}$, $h_i = {G_x}^{c_i}$, $P_i = g^{s_i}$, $Q_i = {g_y}^{s_i}$ for $i = 1, \ldots, 6$ with $G_x = {g_x}^{H(tag)} h_x$. In addition, compute $b_i'' = c_i/a_i$, $b_i' = b_i^*/b_i''$, $w_i = k_i - b_i' e_{ID^*}$ for $i = 1, \ldots, 6$. Then, take random r_0, t_0 from $\mathbb{Z}_q$ and compute $M_1 = g_1^{m_0} g_4^{r_0}$, $M_2 = g_2^{m_0} g_5^{t_0}$, $M_3 = g_3^{m_0} g_6^{r_0+t_0}$. Send $(g_i), (h_i), (P_i), (Q_i), (b_i'), (w_i), (M_i)$.
  3. (Output an honest decommitment.) Receiving a true value m, compute $r = r_0 + \frac{c_1}{c_4}(m_0 - m)$, $t = t_0 + \frac{c_2}{c_5}(m_0 - m)$, and output m, r, t.
- **Extractor** EXT on input $(\tau, \tau^+, view_A)$ does as follows:
  1. For each of the two parallel executions of the subscheme, take necessary items from $view_A$ and set $w = (e_{ID^*}, g_{ID}, \tau, (A_i), (U_i), (b_i), (g_i), (b_i'), (M_i))$. (Note the secret key e_{ID^*} is in $view_A$.)
  2. Invoke the KEA-extractor H^* (in the proof of Claim 4) on $(g, g^+;\ z, w, w^+)$ to get and output m^*.

Fig. 6. EQV and EXT.

B Experiments in the proof of Proposition 3

Experiment $\mathbf{Exp}_0$ on input $m \in \mathbb{Z}_q$ and $z \in \{0,1\}^*$:

1. (Simulate left and right parties for A.) Invoke A on z. Receiving ID^*, ID and tag from A, simulate the left and right parties for A as follows, for each of the two parallel executions of the subscheme.
   (a) (Emulate a public/private-key pair and CRS for A.) Select a random e_{ID^*} from $\mathbb{Z}_q$ and a random element g from G, and set $g_{ID^*} = g^{e_{ID^*}}$. Select a random element g_{ID} from G. Give to A (g_{ID^*}, e_{ID^*}) as A's pair of public/private key and g_{ID} as the right-party's public key. In addition, choose e_c, e_x, d_x, e_y randomly from $\mathbb{Z}_q$, set $g_c = g^{e_c},\ g_x = g^{e_x},\ h_x = g_x^{d_x},\ g_y = g^{e_y}$, and give (g, g_c, g_x, h_x, g_y) as CRS to A.
   (b) (Emulate a first left-party message.) Receiving a request for a first left-party message from A, choose $a_1, \dots, a_6, s_1, \dots, s_6, w_1, \dots, w_6, b'_1, \dots, b'_6$ randomly from $\mathbb{Z}_q^*$ and compute $A_i = g^{a_i} g_c^{s_i},\ U_i = g^{w_i} {g_{ID^*}}^{b'_i}$ for $i = 1, \dots, 6$. Send $A_1, \dots, A_6, U_1, \dots, U_6$ to A.
   (c) (Emulate a first right-party message.) Receiving a first message $tag^*, A_1^*, \dots, A_6^*, U_1^*, \dots, U_6^*$ to the right-party from A, choose $b_1, \dots, b_6$ randomly from ${\mathbb{Z}_q}^*$ and send them to A.
   (d) (Emulate a second left-party message.) Receiving a first message $b_1^*, \dots, b_6^*$ to the left-party from A, compute $b''_i = b_i^*/b'_i,\ c_i = a_i b''_i,\ g_i = g^{c_i},\ h_i = {G_x}^{c_i},\ P_i = g^{s_i},\ Q_i = {g_y}^{s_i}$ for $i = 1, \dots, 6$ with $G_x = {g_x}^{H(tag)} h_x$. Then, take two random r, t from $\mathbb{Z}_q$ and compute $M_1 = g_1^m g_4^r,\ M_2 = g_2^m g_5^t,\ M_3 = g_3^m g_6^{r+t}$. Send $(g_i), (h_i), (P_i), (Q_i), (b'_i), (w_i), (M_i)$ to A.
   (e) (Complete commitment phases.) Receiving a second message $(g_i^*), (h_i^*), (P_i^*), (Q_i^*), (b^{*\prime}{}_i), (w_i^*), (M_i^*)$ to the right-party from A, compute ${b''}_i^* = b_i / {b'}_i^*$ and check

$$\mathrm{DH}(g, G_x^*, g_i^*, h_i^*),\ \mathrm{DH}(g, g_c, P_i^*, Q_i^*),\ U_i^* = {g^{w_i}}^* {g_{ID}}^{{b'}_i^*},\ e(A_i^*, g^{{b''}_i^*}) = e(g_i^*, g_c) e(P_i^*, g_c^{{b''}_i^*})$$

   with $G_x^* = {g_x}^{H(tag^*)} h_x$ for $i = 1, \dots, 6$. If any of them (for any i) is not true, abort with output $\perp$.
2. (Generate output.) Let $view_A$ be the view of A after completing the above commitment phase. If the two values of $m^* = \det(M_1^*, M_2^*, M_3^*)$ determined in the two parallel executions are the same, return $((\tau, \tau^+), view_A, m^*, ((m, r, t), (m, r^+, t^+)))$, otherwise return $\perp$.

Experiment $\mathbf{Exp}_1$ on input $m \in \mathbb{Z}_q$ and $z \in \{0,1\}^*$:

1. (Simulate left and right parties for A.) Invoke A on z. Receiving ID^*, ID and tag from A, simulate the left and right parties for A as follows, for each of the two parallel executions of the subscheme.
   (a) (Emulate a public/private-key pair and CRS for A.) Select a random e_{ID^*} from $\mathbb{Z}_q$ and a random element g from G, and set $g_{ID^*} = g^{e_{ID^*}}$. Select a random element g_{ID} from G. Give to A (g_{ID^*}, e_{ID^*}) as A's pair of public/private key and g_{ID} as the right-party's public key. In addition, choose e_c, e_x, d_x, e_y randomly from $\mathbb{Z}_q$, set $g_c = g^{e_c},\ g_x = g^{e_x},\ h_x = g_x^{d_x},\ g_y = g^{e_y}$, and give (g, g_c, g_x, h_x, g_y) as CRS to A.
   (b) (Emulate a first left-party message.) Receiving a request for a first left-party message from A, choose $a_1, \dots, a_6, s_1, \dots, s_6, \boxed{k_1, \dots, k_6}$ randomly from $\mathbb{Z}_q^*$ and compute $A_i = g^{a_i} g_c^{s_i},\ \boxed{U_i = g^{k_i}}$ for $i = 1, \dots, 6$. Send $A_1, \dots, A_6, U_1, \dots, U_6$ to A.
   (c) (Emulate a first right-party message.) Receiving a first message $tag^*, A_1^*, \dots, A_6^*, U_1^*, \dots, U_6^*$ to the right-party from A, choose $b_1, \dots, b_6$ randomly from ${\mathbb{Z}_q}^*$ and send them to A.
   (d) (Emulate a second left-party message.) Receiving a first message $b_1^*, \dots, b_6^*$ to the left-party from A, $\boxed{\text{select random } c_1, \dots, c_6 \text{ from } {\mathbb{Z}_q}^*}$ and compute $g_i = g^{c_i},\ h_i = {G_x}^{c_i},\ P_i = g^{s_i},\ Q_i = {g_y}^{s_i}$ for $i = 1, \dots, 6$ with $G_x = {g_x}^{H(tag)} h_x$. In additon, compute $\boxed{b''_i = c_i / a_i,\ b'_i = b_i^* / b''_i,\ w_i = k_i - b'_i e_{ID^*}}$ for $i = 1, \dots, 6$, Then, take two random r, t from $\mathbb{Z}_q$ and compute $M_1 = g_1^m g_4^r,\ M_2 = g_2^m g_5^t,\ M_3 = g_3^m g_6^{r+t}$. Send $(g_i), (h_i), (P_i), (Q_i), (b'_i), (w_i), (M_i)$ to A.
   (e) (Complete commitment phases.) Receiving a second message $(g_i^*), (h_i^*), (P_i^*), (Q_i^*), (b^{*\prime}{}_i), (w_i^*), (M_i^*)$ to the right-party from A, compute ${b''}_i^* = b_i / {b'}_i^*$ and check

$$\mathrm{DH}(g, G_x^*, g_i^*, h_i^*),\ \mathrm{DH}(g, g_c, P_i^*, Q_i^*),\ U_i^* = {g^{w_i}}^* {g_{ID}}^{{b'}_i^*},\ e(A_i^*, g^{{b''}_i^*}) = e(g_i^*, g_c) e(P_i^*, g_c^{{b''}_i^*})$$

   with $G_x^* = {g_x}^{H(tag^*)} h_x$ for $i = 1, \dots, 6$. If any of them (for any i) is not true, abort with output $\perp$.
2. (Generate output.) Let $view_A$ be the view of A after completing the above commitment phase. If the two values of $m^* = \det(M_1^*, M_2^*, M_3^*)$ determined in the two parallel executions are the same, return $((\tau, \tau^+), view_A, m^*, ((m, r, t), (m, r^+, t^+)))$, otherwise return $\perp$.

Experiment $\mathbf{Exp}_2$ on input $m \in \mathbb{Z}_q$ and $z \in \{0,1\}^*$:

1. (Simulate left and right parties for A.) Invoke A on z. Receiving ID^*, ID and tag from A, simulate the left and right parties for A as follows, for each of the two parallel executions of the subscheme.
   (a) (Emulate a public/private-key pair and CRS for A.) Select a random e_{ID^*} from $\mathbb{Z}_q$ and a random element g from G, and set $g_{ID^*} = g^{e_{ID^*}}$. Select a random element g_{ID} from G. Give to A (g_{ID^*}, e_{ID^*}) as A's pair of public/private key and g_{ID} as the right-party's public key. In addition, choose e_c, e_x, d_x, e_y randomly from $\mathbb{Z}_q$, set $g_c = g^{e_c},\ g_x = g^{e_x},\ h_x = g_x^{d_x},\ g_y = g^{e_y}$, and give (g, g_c, g_x, h_x, g_y) as CRS to A.
   (b) (Emulate a first left-party message.) Receiving a request for a first left-party message from A, choose $a_1, \ldots, a_6, s_1, \ldots, s_6, k_1, \ldots, k_6$ randomly from $\mathbb{Z}_q^*$ and compute $A_i = g^{a_i} g_c^{s_i}$, $U_i = g^{k_i}$ for $i = 1, \ldots, 6$. Send $A_1, \ldots, A_6, U_1, \ldots, U_6$ to A.
   (c) (Emulate a first right-party message.) Receiving a first message $tag^*, A_1^*, \ldots, A_6^*, U_1^*, \ldots, U_6^*$ to the right-party from A, choose $b_1, \ldots, b_6$ randomly from ${\mathbb{Z}_q}^*$ and send them to A.
   (d) (Emulate a second left-party message.) Receiving a first message $b_1^*, \ldots, b_6^*$ to the left-party from A, select random $c_1, \ldots, c_6$ from $\mathbb{Z}_q^*$ so that (g^{c_i}) becomes a linear tuple and compute $g_i = g^{c_i},\ h_i = {G_x}^{c_i},\ P_i = g^{s_i},\ Q_i = {g_y}^{s_i}$ for $i = 1, \ldots, 6$ with $G_x = {g_x}^{H(tag)} h_x$. In additon, compute $b_i'' = c_i/a_i,\ b_i' = b_i^*/b_i'',\ w_i = k_i - b_i' e_{ID^*}$ for $i = 1, \ldots, 6$. Then, take two random r, t from $\mathbb{Z}_q$ and compute $M_1 = g_1^m g_4^r,\ M_2 = g_2^m g_5^t,\ M_3 = g_3^m g_6^{r+t}$. Send $(g_i), (h_i), (P_i), (Q_i), (b_i'), (w_i), (M_i)$ to A.
   (e) (Complete commitment phases.) Receiving a second message $(g_i^*), (h_i^*), (P_i^*), (Q_i^*), (b^{*\prime}{}_i), (w_i^*), (M_i^*)$ to the right-party from A, compute ${b''}_i^* = b_i / {b'}_i^*$ and check

$$\mathrm{DH}(g, G_x^*, g_i^*, h_i^*),\ \mathrm{DH}(g, g_c, P_i^*, Q_i^*),\ U_i^* = {g^{w_i}}^* {g_{ID}}^{{b'}_i^*},\ e(A_i^*, g^{{b''}_i^*}) = e(g_i^*, g_c) e(P_i^*, g_c^{{b''}_i^*})$$

   with $G_x^* = g_x^{H(tag^*)} h_x$ for $i = 1, \ldots, 6$. If any of them (for any i) is not true, abort with output $\perp$.
2. (Generate output.) Let $view_A$ be the view of A after completing the above commitment phase. If the two values of $m^* = \det(M_1^*, M_2^*, M_3^*)$ determined in the two parallel executions are the same, return $((\tau, \tau^+), view_A, m^*, ((m, r, t), (m, r^+, t^+)))$, otherwise return $\perp$.

Experiment $\mathbf{Exp}_3$ on input $m \in \mathbb{Z}_q$ and $z \in \{0,1\}^*$:

1. (Simulate left and right parties for A.) Invoke A on z. Receiving ID^*, ID and tag from A, simulate the left and right parties for A as follows, for each of the two parallel executions of the subscheme.
   (a) (Emulate a public/private-key pair and CRS for A.) Select a random e_{ID^*} from $\mathbb{Z}_q$ and a random element g from G, and set $g_{ID^*} = g^{e_{ID^*}}$. Select a random element g_{ID} from G. Give to A (g_{ID^*}, e_{ID^*}) as A's pair of public/private key and g_{ID} as the right-party's public key. In addition, choose e_c, e_x, d_x, e_y randomly from $\mathbb{Z}_q$, set $g_c = g^{e_c},\ g_x = g^{e_x},\ h_x = g_x^{d_x},\ g_y = g^{e_y}$, and give (g, g_c, g_x, h_x, g_y) as CRS to A.
   (b) (Emulate a first left-party message.) Receiving a request for a first left-party message from A, choose $a_1, \ldots, a_6, s_1, \ldots, s_6, k_1, \ldots, k_6$ randomly from $\mathbb{Z}_q^*$ and compute $A_i = g^{a_i} g_c^{s_i}$, $U_i = g^{k_i}$ for $i = 1, \ldots, 6$. Send $A_1, \ldots, A_6, U_1, \ldots, U_6$ to A.
   (c) (Emulate a first right-party message.) Receiving a first message $tag^*, A_1^*, \ldots, A_6^*, U_1^*, \ldots, U_6^*$ to the right-party from A, choose $b_1, \ldots, b_6$ randomly from ${\mathbb{Z}_q}^*$ and send them to A.
   (d) (Emulate a second left-party message.) Receiving a first message $b_1^*, \ldots, b_6^*$ to the left-party from A, select random $c_1, \ldots, c_6$ from $\mathbb{Z}_q^*$ so that (g^{c_i}) becomes a linear tuple and compute $g_i = g^{c_i},\ h_i = {G_x}^{c_i},\ P_i = g^{s_i},\ Q_i = {g_y}^{s_i}$ for $i = 1, \ldots, 6$ with $G_x = {g_x}^{H(tag)} h_x$. In additon, compute $b_i'' = c_i/a_i,\ b_i' = b_i^*/b_i'',\ w_i = k_i - b_i' e_{ID^*}$ for $i = 1, \ldots, 6$. Then, take two random r_0, t_0 from $\mathbb{Z}_q$ and compute $M_1 = g_1^{m_0} g_4^{r_0},\ M_2 = g_2^{m_0} g_5^{t_0},\ M_3 = g_3^{m_0} g_6^{r_0 + t_0}$ with an arbitrary dummy value $m_0 \in mathbbZ_q$. Send $(g_i), (h_i), (P_i), (Q_i), (b_i'), (w_i), (M_i)$ to A. Compute $r = r_0 + \frac{c_1}{c_4}(m_0 - m),\ t = t_0 + \frac{c_2}{c_5}(m_0 - m)$.
   (e) (Complete commitment phases.) Receiving a second message $(g_i^*), (h_i^*), (P_i^*), (Q_i^*), (b^{*\prime}{}_i), (w_i^*), (M_i^*)$ to the right-party from A, compute ${b''}_i^* = b_i / {b'}_i^*$ and check

$$\mathrm{DH}(g, G_x^*, g_i^*, h_i^*),\ \mathrm{DH}(g, g_c, P_i^*, Q_i^*),\ U_i^* = {g^{w_i}}^* {g_{ID}}^{{b'}_i^*},\ e(A_i^*, g^{{b''}_i^*}) = e(g_i^*, g_c) e(P_i^*, g_c^{{b''}_i^*})$$

   with $G_x^* = g_x^{H(tag^*)} h_x$ for $i = 1, \ldots, 6$. If any of them (for any i) is not true, abort with output $\perp$. Else let $w = (e_{ID^*}, g_{ID}, \tau, (A_i), (U_i), (b_i), (g_i), (b_i'), (M_i))$.
2. (Generate output.) Let g and g^+ be the first elements of CRSs in the two executions of the subscheme. Invoke KEA-extractor H^* on $(g, g^+;\ \ z, w, w^+)$ to get m^* and return $((\tau, \tau^+), view_A, m^*, ((m, r, t), (m, r^+, t^+)))$.