CCA Secure Threshold KEM Scheme Against Adaptive Corruption Attacks in Standard Model

Yuanju Gan[a,b], Licheng Wang[a], Jianhua Yan[a], Yixian Yang[a]
[a] Information Security Center, Beijing University of Posts and Telecommunications, Beijing 100876, China
Email: {ganyj66,wanglc, yjh, yxyang}@bupt.edu.cn
[b] School of Information, Guangdong ocean University, Zhanjiang 524088,China
Email: ganyj66@163.com

***Abstract*— Most threshold key encapsulation mechanisms (KEM) have been studied in a weak model–static corruption model or random oracle model. In this paper, we propose a threshold KEM scheme with provable security based on the bilinear groups of composite order in the standard model. We use a direct construction from Boyen-Mei-Waters' KEM scheme to obtain a threshold KEM scheme that can withstand adaptive chosen ciphertext attacks (CCA) and adaptive corruption attacks. However, to achieve a higher security level, our construction does not increase overall additional size of ciphertext compare to other schemes.**

***Index Terms*— Key encapsulation mechanisms; Adaptive corruption attacks; Chosen ciphertext attack; Bilinear groups of composite order**

I. Introduction

In 1998, Cramer and Shoup [1] proposed the first practical public key encryption (PKE) scheme whose security against adaptive chosen ciphertext attacks (C-CA) could be proven without depending on the random oracle model. Security against CCA is now commonly accepted as the standard security notion for public key encryption schemes. In a threshold public-key encryption (TPKE) system [2], [3], each of n users holds a secret decryption key corresponding to a public key, a message is encrypted and sent to a group of decryption users, and the ciphertext can be decrypted only if at least t of decryption users (where t is the threshold) in the authorized set cooperate. Below this threshold, no information about the plaintext is leaked, even if the number of the authorized users was corrupted up to t-1, which is crucial in all applications and situations where one cannot fully trust a single person, but possibly a group of individuals. The security notions of threshold encryption are very similar to those of public-key encryption in that the notion of indistinguishability against chosen ciphertext attacks (IND-CCA) in public key encryption corresponds to the notion of indistinguishability against chosen ciphertext attacks in threshold encryption (IND-TCCA). However, the static adversary model or adaptive adversary model is a special security notion in threshold public-key encryption. In the static corruption model the adversary fixes the players that will be corrupted before the protocol starts, while in the adaptive corruption model, the adversary chooses which players to corrupt at any time and based on any information it sees during the protocol. Obviously, the notion of indistinguishability against static corruption attacks and chosen ciphertext attacks in threshold encryption(IND-SCA-TCCA) [4] is weaker than the notion of indistinguishability against adaptive(or named dynamic) corruption attacks and chosen ciphertext attacks in threshold encryption (IND-ACA-TCCA) [5]–[8].

Instead of providing the full functionality of the public-key encryption scheme, in many applications the communication between a sender and receiver only needs a temporary session key to encrypt a message. The key encapsulation mechanism (KEM) [9]–[11] is used to transmit a randomly encrypted key from a sender to a designated receiver instead of a message. A sender runs an encapsulation algorithm to produce a random session key together with a corresponding ciphertext. This ciphertext is sent to the receiver, which can uniquely reconstruct the session key by using its secret key. In the end, both parties share a common random session key. The KEM in the threshold settings is that: each of n users holds a secret decryption key corresponding to a public key; a session key is encrypted and sent to a group of decryption users; and the ciphertext can be decrypted only if at least t decryption users in the authorized set cooperate.

The security notions of threshold KEM (TKEM) are similar to those of threshold encryption. The strongest notion is indistinguishability against adaptive corruption attacks and chosen ciphertext attacks.

In 2005, Boyen-Mei-Waters [10] proposed an IND-CCA-TKEM scheme in the standard model. However, the security reduction of [10] is loose. therefore, in the same secure level, the size of the system secure parameter of loose secure reduction will be much larger than that of tight secure reduction. In 2007, based on RSA problem, Takeru et al. [12] proposed a TKEM scheme against IND-CCA with tight secure reduction. However, the secure model of [12] is random oracle model [13].

Historically, most threshold key encapsulation mech-

This work was supported by the National Natural Science Foundation of China (NSFC) (60973159, 61070251, 61103198, 61121061, 61272534) and the NSFC A3 Foresight Program 61161140320.

doi:10.4304/jsw.8.5.1245-1253

anisms [10], [12], [14] have been studied in a static corruption model, that is, a adversary chooses which users it wants to corrupt before the scheme is setup. However, in adaptive corruption model, the a adversary could choose which users it wants to corrupt at any time. So, the static corruption model is weaker than the adaptive corruption model. In 2011, Libert and Yung [15] use the Lewko-Waters [16] dual encryption approach and bilinear group with composite orders to design a threshold decryption scheme that is simultaneously chosen-ciphertext secure under adaptive corruptions and non-interactive. However, to achieve CCA security, Libert and Yung use a one-time strong signature, so the ciphertext of their scheme is longer than schemes without one.

In this paper, based on Boyen-Mei-Waters' TKEM, and Libert and Yung's threshold decryption scheme, we construct a robust CCA threshold KEM scheme against adaptive corruption attacks with tight secure reduction in the standard model. To the best of our knowledge, this is the first threshold KEM scheme that can withstand adaptive corruption attacks and chosen ciphertext attacks in the standard model.

II. Preliminaries

A. Bilinear Group with Composite Orders and Related Cryptographic Assumptions

Let $\mathbb{G}$ and $\mathbb{G}_{\mathbb{T}}$ be two cyclic groups of order $N = p_1p_2p_3$ (where p_1, p_2, p_3 are distinct primes). A bilinear map $e(\cdot,\cdot)$ is a map $\mathbb{G} \times \mathbb{G} \rightarrow \mathbb{G}_{\mathbb{T}}$ such that for any generator $g, h \in \mathbb{G}$ and random $\alpha, \beta \in \mathbb{Z}_N$, it satisfies the following properties:

- **Bilinearity:** $e(g^{\alpha}, g^{\beta}) = e(g,g)^{\alpha\beta}$.
- **Non-degeneracy:** If $e(g,h) = 1_{\mathbb{G}_{\mathbb{T}}}$, for all $h \in \mathbb{G}$, then $g = 1_{\mathbb{G}}$.
- **Orthogonality:** Let G_{p_1}, G_{p_2}, and G_{p_3} denote the subgroups of order p_1, p_2 and p_3 in $\mathbb{G}$ respectively. $e(h_i, h_j)$ is the identity element in $\mathbb{G}_{\mathbb{T}}$, for any $h_i \in \mathbb{G}_{p_i}$ and $h_j \in \mathbb{G}_{p_j}$($i \neq j$), that is $e(h_i, h_j) = 1_{\mathbb{G}_{\mathbb{T}}}(i \neq j)$.

For each $i \in \{1,2,3\}$, the notation $\mathbb{G}_{p_i}$ is the subgroup of order p_i. For all distinct $i, j \in \{1,2,3\}$, notation $\mathbb{G}_{p_ip_j}$ is the subgroup of order p_ip_j.

The following assumptions on a bilinear group with composite orders will be used in this paper. For more details please refer to [16].

Assumption 1 ([16]): Given a description of $(N = p_1p_2p_3, \mathbb{G}, \mathbb{G}_{\mathbb{T}}, e(\cdot,\cdot))$, as well as $g \in_R \mathbb{G}_{p_1}, X_3 \in_R \mathbb{G}_{p_3}$ and $T \in_R \mathbb{G}$, it is infeasible to efficiently decide whether $T \in \mathbb{G}_{p_1p_2}$ or $T \in \mathbb{G}_{p_1}$.

Assumption 2 ([16]): Given a description of $(N = p_1p_2p_3,\ \mathbb{G},\ \mathbb{G}_{\mathbb{T}}, e(\cdot,\cdot))$, a set of group elements (g, X_1X_2, Z_3, Y_2Y_3) and $T \in_R \mathbb{G}$, where $(g, X_1) \in_R \mathbb{G}_{p_1}^2, (X_2, Y_2) \in_R \mathbb{G}_{p_2}^2$, and $(Y_3, Z_3) \in_R \mathbb{G}_{p_3}^2$, it is hard to efficiently decide whether $T \in \mathbb{G}_{p_1p_3}$.

Assumption 3 ([16]): Given a description of $(N = p_1p_2p_3, \mathbb{G}, \mathbb{G}_{\mathbb{T}}, e(\cdot,\cdot))$, a set of group elements $(g, g^{\alpha}X_2$, $X_3, g^{\beta}Y_2,\ Z_2)$ and $\mathcal{T} \in_R \mathbb{G}_{\mathbb{T}}$, where $g \in_R \mathbb{G}_{p_1}, (X_2, Y_2$, $Z_2) \in_R \mathbb{G}_{p_2}^3, X_3 \in_R \mathbb{G}_{p_3}$ and $(\alpha, \beta) \in_R \mathbb{Z}_N^2$ it is infeasible to efficiently decide whether $\mathcal{T} = e(g,g)^{\alpha\beta}$.

Lemma 1 (Lemma 1 in [16]): If an algorithm can produce a nontrivial factor of N, then it can break Assumption 1 or Assumption 2.

B. Definition of (t,n)-Threshold KEM Scheme

Let $\mathcal{P} = (P_1, \cdots, P_n)$ be a set of n participants. A sender wants to send a session key K to $\mathcal{P}$ that any t participants can recover session key K, while $t-1$ participants cannot acquire any information about session key K. A (t,n)-threshold KEM scheme consists of the following six algorithms:

- **Setup**(Λ, t, n): Takes as input a security parameter Λ, decryption threshold t, and a number of decryption participants n. It outputs a set of parameters $(PK, \mathbf{SK}, \mathbf{VK})$, where PK is the public key, $\mathbf{SK} = (SK_1, \ldots, SK_n)$ and $\mathbf{VK} = (VK_1, \ldots, VK_n)$ are the corresponding decryption keys and verification keys, respectively. The ith participant is given the decryption key share (i, SK_i).
- **Encapsulate**(PK): The algorithm randomly selects a secret $k \in \mathbb{Z}_N$, then outputs the ciphertext C and the session key K.
- **CiphertextVerify**(PK, C): Takes as input the public key PK and ciphertext C. It checks whether C is a valid ciphertext with respect to PK.
- **PartialDecapsulate(**PK, SK_i, C**):** Takes as input the public key PK, a ciphertext C, P_i's decryption key SK_i. It outputs a partial decapsulation share μ_i of the ciphertext C, or a special symbol $(i, \perp)$.
- **ShareVerify(**PK, VK, C, μ_i**):** Takes as input the public key PK, verification keys VK, as well as a ciphertext C and partial decapsulation share μ_i. It checks whether μ_i is a valid partial decapsulation share with respect to VK. It outputs valid or invalid.
- **Reconstruct(**$PK, \mathbf{VK}, C, \Omega$**):** Takes as input the public key PK, verification keys $\mathbf{VK}$, as well as a ciphertext C, and a list of t partial decapsulation shares, denoted by $\Omega = (\mu_1, \ldots, \mu_t)$, without loss of generality. It outputs a session key K or $\perp$.

Let (PK,**SK**,**VK**) be the output of the *Setup*(n, t, λ). We require the following two consistency properties:

1) For any ciphertext C generated by the *Encapsulation*(PK) algorithm, if μ_i is generated by the *PartialDecapsulate*(PK, SK_i, C), where SK_i is P_i's decryption key share, then ShareVerify(PK, VK_i, C, μ_i) = valid.
2) If C is the output of the *Encapsulation(PK)* algorithm and $\Omega = (\mu_1, \ldots, \mu_t)$ is a list of t distinct partial decapsulation shares μ_i, where $\mu_i =$ *PartialDecapsulate*(PK, SK_i, C), then we require that *Reconstruction*$(PK, \mathbf{VK}, C, \Omega)$=$K$.

C. Security Model

For any ciphertext C associated with a session key K, any collusion for which fewer than t participants cannot learn any information about the session key K. Following [4] [10] [14], we further formally define the security of threshold KEM against IND-ACA-TCCA (adaptive corruption attacks, chosen ciphertext attacks), under the classical semantic security notion, and using the following game between an adversary $\mathcal{A}$ and challenger $\mathcal{C}$. Both are given as input n, t, and a security parameter Λ.

- **Init:** The challenger $\mathcal{C}$ runs Setup(n, t, Λ) algorithm to obtain the set of parameters PK, $\mathbf{SK} = (SK_1, \ldots, SK_n)$, and $\mathbf{VK} = (VK_1, \ldots, VK_n)$. It gives PK, $\mathbf{VK}$ to the adversary $\mathcal{A}$.
- **Phase 1:** The adversary $\mathcal{A}$ adaptively issues the following queries:
  Corruption query: The adversary $\mathcal{A}$ adaptively issues a decryption key share query of a participant depending on the results of previous attacks. If the adversary $\mathcal{A}$ wants to query the ith decryption key, the challenger $\mathcal{C}$ forwards the corresponding decryption key SK_i to adversary $\mathcal{A}$. No more than $t-1$ decryption key shares can be obtained by $\mathcal{A}$ in the whole game.
  DecapsulationShare query: The adversary $\mathcal{A}$ adaptively issues DecapsulationShare query with (P_i, C), where $i \in \{1, \ldots, n\}$. The challenger $\mathcal{C}$ runs the PartialDecapsulate algorithm with C and SK_i, and forwards the resulting partial decapsulation share of the P_i to adversary $\mathcal{A}$.
- **Challenge:** The challenger $\mathcal{C}$ picks a random bit $\delta \in \{0, 1\}$ and runs the *Encapsulate* algorithm to obtain (C^*, K_0), and randomly chooses an ephemeral key K_1. Challenger $\mathcal{C}$ then gives (K_δ, C^*) to the adversary $\mathcal{A}$.
- **Phase 2:** The adversary $\mathcal{A}$ makes further queries as in Phase 1, but it is not allowed to make DecapsulationShare query on the challenge C^*.
- **Guess:** Finally, the adversary $\mathcal{A}$ outputs a guess $\delta' \in \{0, 1\}$ and wins the game if $\delta = \delta'$.

Let $Adv_{\mathcal{A},n,t}^{IND-ACA-TCCA}$ denote the probability that $\mathcal{A}$ wins the game when the challenger $\mathcal{C}$ and adversary $\mathcal{A}$ are given *n,t* as input.

We say that a TKEM is CCA security if for any *n* and *t*, where $0 < \mathrm{t} \leq \mathrm{n}$, and the advantage of any probabilistic polynomial-time (PPT) adversary $\mathcal{A}$, the advantage $Adv_{\mathcal{A},n,t}^{IND-ACA-TCCA}(\Lambda) = |\Pr[\delta' = \delta] - 1/2|$ is negligible with Λ.

III. The Proposed scheme

The algorithms of our (t, n)-threshold KEM scheme are specified as follows:

- **Setup**(Λ, t, n). Given the parameter λ,t,n, this algorithm does the following:
  1) Select bilinear groups $(\mathbb{G}, \mathbb{G}_\mathbb{T})$ of order $N = p_1p_2p_3$ (where p_1, p_2, p_3 are distinct primes and $p_1, p_2, p_3 > 2^\Lambda$), a bilinear map $e : \mathbb{G} \times \mathbb{G} \to \mathbb{G}_\mathbb{T}$;
  2) Select generators $(g, h, u, v) \in_R \mathbb{G}_{p_1}^4, X_{p_3} \in_R \mathbb{G}_{p_3}$;
  3) Select a collision-resistance hash function $H : \mathbb{G} \to \mathbb{Z}_N$;
  4) Pick a random degree $t-1$ polynomial $f(x) = \sum_{j=0}^{t-1} a_j x^j$ (where t is the value of threshold, and $a_j \in \mathbb{Z}_N$, for $j = 0, \cdots, t-1$, and $\alpha = a_0 \neq 0$), and compute the decryption key share $SK_i = h^{f(i)} \cdot Z_{3,i}$ and verification key $VK_i = e(SK_i, g) = e(g, h)^{f(i)}$, for $i = 1, \ldots, n$, where $Z_{3,i}$ is selected in $\mathbb{G}_{p_3}$ at random. (Random elements of $\mathbb{G}_{p_3}$ can be obtained taking a generator of X_{p_3} and raising it to random exponent modulo N);
  5) Publish the public key to be $PK = (\mathbb{G}, \mathbb{G}_\mathbb{T}, N, g, u, v, X_{p_3}, e(\cdot, \cdot), H, e(g, h)^\alpha)$ and the verification keys $\mathbf{VK} = (VK_1, \ldots, VK_n)$ on the system bulletin board (BB). The decryption key SK_i is privately given to P_i, for $i = 1, \ldots, n$;
  6) The secret parameters are $(h, \alpha, p_1, p_2, p_3)$.
- **Encapsulate**(PK). Given the PK, this algorithm first picks $k \in_R \mathbb{Z}_N$ and computes

$$c_1 = g^k,$$
$$c_2 = (u^\tau v)^k,$$

  where $\tau = H(c_1)$. The complete encapsulated key, C is the two group elements (c_1, c_2). The session key, $K = e(g, h)^{\alpha k}$, is calculated and kept secretly by the sender.
- **CiphertextVerify**(PK, C). Given $C = (c_1, c_2)$ and PK, any verifier first randomly picks two distinct elements $g_3, g_3' \in \mathbb{G}_{p_3}$ and checks whether or not

$$e(c_2, gg_3) \stackrel{?}{=} e(c_1, (u^\tau v)g_3'), \tag{1}$$

  is correct, where $\tau = H(c_1)$.
- **PartialDecapsulate**(PK, C, SK_i). Given the public key PK and an encapsulated key $C = (c_1, c_2)$, the participant P_i returns 0 if CiphertextVerify(PK, C) is not correct. Otherwise, it randomly picks $\beta_i \in \mathbb{Z}_N, (W_{3i}, W_{3i}') \in \mathbb{G}_{p_3}^2$, and computes

$$\mu_{i1} = SK_i \cdot (u^\tau v)^{\beta_i} \cdot W_{3i},$$
$$\mu_{i2} = g^{\beta_i} W_{3i}',$$

  where $\tau = H(c_1)$. Then participant P_i can send $\mu_i = \{\mu_{i1}, \mu_{i2}\}$ to the combiner through secure channels.
- **ShareVerify**(PK, VK, C, μ_i). The combiner verifies whether $\mu_i = \{\mu_{i1}, \mu_{i2}\}$ generated by the ith participant is valid, as follows. It first computes $\tau = H(c_1)$, then checks whether or not

$$e(\mu_{i1}, g) = VK_i \cdot e(u^\tau v, \mu_{i2}), \tag{2}$$

  is correct. If so, the algorithm outputs valid. Otherwise it outputs invalid.
- **Reconstruct**(PK, VK, C, Ω). The session key is reconstructed from $\Omega = (\mu_1, \ldots, \mu_t)$, a list of t partial decapsulation shares of an encapsulated key $C = (c_1, c_2)$, as follows. The combiner first verifies

that C and t partial decapsulation shares are valid, then computes

$$d_1 = \prod_{i=1}^{t} \mu_{i1}^{\lambda_i},$$
$$d_2 = \prod_{i=1}^{t} \mu_{i2}^{\lambda_i},$$

where $\lambda_i = \prod_{j=1, j\neq i}^{t} \frac{-j}{i-j}$. Finally, the combiner uses (c_1, c_2, d_1, d_2) to reconstruct session key K as follows:

$$K = \frac{e(c_1, d_1)}{e(c_2, d_2)}. \quad (3)$$

IV. Security Analysis

A. Correctness

The consistency of equation (1) is given by

$$\begin{aligned} e(c_2, gg_3) &= e((u^\tau v)^k, gg_3) \\ &= e((u^\tau v)^k, g) e((u^\tau v)^k, g_3), \\ e(c_1, (u^\tau v) g_3') &= e(g^k, (u^\tau v) g_3') \\ &= e(g^k, (u^\tau v)) e(g^k, g_3'), \end{aligned}$$

and $e((u^\tau v)^k, g_3) = 1_{\mathbb{G}_\mathbb{T}} = e(g^k, g_3')$.

The consistency of the equation (2) is given by

$$\begin{aligned} e(\mu_{i1}, g) &= e(SK_i \cdot (u^\tau v)^{\beta_i} \cdot W_{3i}, g) \\ &= e(SK_i, g) \cdot e((u^\tau v)^{\beta_i}, g) \cdot e(W_{3i}, g) \\ &= VK_i \cdot e((u^\tau v), g^{\beta_i}) \\ &= VK_i \cdot e(u^\tau v, g^{\beta_i}) \cdot e(u^\tau v, W_{3i}') \\ &= VK_i \cdot e(u^\tau v, g^{\beta_i} W_{3i}') \\ &= VK_i \cdot e(u^\tau v, \mu_{i2}). \end{aligned}$$

The consistency of the equation (3) is given by

$$\begin{aligned} d_1 &= \prod_{i=1}^{t} \mu_{i1}^{\lambda_i} \\ &= \prod_{i=1}^{t} (h^{f(i)} Z_{3,i} (u^\tau v)^{\beta_i} \cdot W_{3i})^{\lambda_i} \\ &= \prod_{i=1}^{t} \left(h^{f(i)} (u^\tau v)^{\beta_i}\right)^{\lambda_i} \prod_{i=1}^{t} (Z_{3,i} W_{3i})^{\lambda_i} \\ &= \prod_{i=1}^{t} h^{f(i)\lambda_i} \prod_{i=1}^{t} (u^\tau v)^{\beta_i \lambda_i} \cdot R_{3i}' \\ &= h^\alpha \prod_{i=1}^{t} (u^\tau v)^{\beta_i \lambda_i} \cdot R_{3i}', \end{aligned}$$

and

$$d_2 = \prod_{i=1}^{t} \mu_{i2}^{\lambda_i} = \prod_{i=1}^{t} (g^{\beta_i} W_{3i}')^{\lambda_i} = \prod_{i=1}^{t} (g)^{\beta_i \lambda_i} \cdot R_{3i}'',$$

in which $R_{3i}' = \prod_{i=1}^{t} (Z_{3,i} W_{3i})^{\lambda_i}, R_{3i}'' = \prod_{i=1}^{t} (W_{3i}')^{\lambda_i}$, and thus

$$\begin{aligned} \frac{e(c_1, d_1)}{e(c_2, d_2)} &= \frac{e(g^k, h^\alpha \prod_{i=1}^{t} (u^\tau v)^{\beta_i \lambda_i} R_{3i}')}{e\left((u^\tau v)^k, \prod_{i=1}^{t} g^{\beta_i \lambda_i} \cdot R_{3i}''\right)} \\ &= \frac{e(g^k, h^\alpha) e(g^k, \prod_{i=1}^{t} (u^\tau v)^{\beta_i \lambda_i})}{e((u^\tau v)^k, \prod_{i=1}^{t} g^{\beta_i \lambda_i})} \\ &= \frac{e(g^k, h^\alpha) e(g^k, (u^\tau v)^{\sum_{i=1}^{t} \beta_i \lambda_i})}{e((u^\tau v)^k, g^{\sum_{i=1}^{t} \beta_i \lambda_i})} \\ &= e(g^k, h^\alpha) = K. \end{aligned}$$

V. Security

Theorem 1: The scheme is IND-ACA-TCCA secure against adaptive corruptions attacks if Assumption 1, Assumption 2, and Assumption 3 hold simultaneously, and H is a collision-resistant hash function in the standard model.

Proof: We prove the security by a hybrid argument using a sequence of games. The first game, Game_0, is basically identical to the IND-ACA-TCCA game in the standard model, and the adversary $\mathcal{A}$'s advantage is defined accordingly. In the last Game, Game_6, the adversary will still have to guess a given bit. But in this last game, the challenger $\mathcal{C}$ first performs the **Encapsulate** algorithm to acquire (K, C^*), and selects a random bit $\delta \in \{0, 1\}$ and two random session keys K_0, K_1, then sends (C^*, K_δ) to the adversary $\mathcal{A}$. So $|\Pr[\delta' = \delta]|$ must be exactly 1/2. To go from the first game to the last, we define various intermediate games. According to the method of proofing sequence of games [17], each Game_i must be very similar to Game_{i-1}, that is, the advantage of $\mathcal{A}$ in Game_i will be bounded away from its advantage in Game_{i-1} by at most a negligible quantity. Let g_2 and g_3 denote a generator of the subgroup $\mathbb{G}_{p_2}$ and $\mathbb{G}_{p_3}$, respectively.

- **Game_0**: We now define a game, Game_0, that is an interactive computation between a challenger $\mathcal{C}$ and adversary $\mathcal{A}$. This game is simply the usual IND-ACA-TCCA game, in which $\mathcal{C}$ provides the adversary's environment.
  - **Init** : $\mathcal{C}$ runs the **Setup** algorithm to obtain the description of $PK = (\mathbb{G}, \mathbb{G}_\mathbb{T}, N, g, u, v, X_{p_3}, e(\cdot,\cdot), H, e(g, h)^\alpha)$, $MSK = (h, p_1, p_2, p_3)$, and picks a polynomial $f(x) = \alpha + a_1 x + \ldots + a_{t-1} x^{t-1}$ to compute P_i's decryption key share $SK_i = h^{f(i)} Z_{3,i}$ and verification key $VK_i = e(g, SK_i)$, for $i = 1, \ldots, n$, where $Z_{3,i} \in_R \mathbb{G}_{p_3}$. $\mathcal{C}$ gives the public key PK and verification key $\mathbf{VK} = \{VK_1, \ldots, VK_n\}$ to $\mathcal{A}$.
  - **Phase** 1: $\mathcal{A}$ can adaptively issue a "Corruption query" and "DecapsulationShare query."
    Corruption query : If $\mathcal{A}$ wants to corrupt P_i, $\mathcal{C}$ just gives the decryption key share SK_i to $\mathcal{A}$. No more than $t-1$ decryption key shares can be obtained by $\mathcal{A}$ in the game_0.
    DecapsulationShare query: Received $(P_i, C = (c_1, c_2))$ from $\mathcal{A}$, the challenger $\mathcal{C}$ checks whether C is well-formed according to the equality (1). If so, $\mathcal{C}$ randomly picks $\beta_i \in \mathbb{Z}_N, (W_{3i}, W_{3i}') \in G_{p_3}^2$, computes $\mu_{i1} = SK_i \cdot (u^\tau v)^{\beta_i} \cdot W_{3i}$, $\mu_{i2} = g^{\beta_i} W_{3i}'$, where $\tau = H(c_1)$, and sends the partial decapsulation share $\mu_i = (\mu_{i1}, \mu_{i2})$ to $\mathcal{A}$. Otherwise, $\mathcal{C}$ gives a random value to $\mathcal{A}$.
  - **Challenge:** Once $\mathcal{A}$ ends the Phase 1, $\mathcal{C}$ can form the following challenge information. $\mathcal{C}$ randomly selects $k \in \mathbb{Z}_N$, and computes $C^* = (c_1^* = g^k, c_2^* = (u^\tau v)^k), K_0 = e(g, h)^{\alpha k}$, where $\tau = H(c_1)$. With this, $\mathcal{C}$ now selects a random

bit $\delta \in \{0,1\}$ and a random session key K_1. $\mathcal{C}$ then sends (C^*, K_δ) to $\mathcal{A}$.

– **Phase** 2: $\mathcal{A}$ continues to issue further **Corruption** query and **DecapsulationShare** query as in phase 1, but it is not allowed to make DecapsulationShare query on the challenge C^*.

– **Guess**: Eventually, $\mathcal{A}$ outputs a guess bit $\delta' \in \{0,1\}$ for δ. Since Game$_0$ is identical to the IND-ACA-TCCA game, we have

$$ADV_{\mathcal{A}}^{IND-ACA-TCCA}(\lambda) = |\Pr[\delta' = \delta] - 1/2|$$

and our goal is to prove that this quantity is negligible.

- **Game**$_1$:This is the same as Game$_1$, except that the challenger will reject all decryption queries that $c_1 \neq c_1^*$ and $H(c_1) = H(c_1^*)$.
- **Game**$_2$: This is identical to Game$_2$, except the challenger will refuse all decryption queries that $H(c_1) \neq H(c_1^*)$ and $H(c_1) = H(c_1^*) \pmod{p_2}$.
- **Game**$_3$: This is the same as Game$_2$, except that in the phase 2, $\mathcal{C}$ will abort if $\mathcal{A}$ manages to make PartialDecapsulate query $(P_i, C = (c_1, c_2))$ such that the $C = (c_1, c_2)$ can pass the **CiphertextVerify** algorithm and for which $c_1 = c_1^*$ and $c_2 \neq c_2^*$.
- **Game**$_4$: This is the same as Game$_3$, with one difference in the Challenge phase that $\mathcal{C}$ randomly selects $(k, \omega, \zeta) \in \mathbb{Z}_N^3$ and generates the challenge ciphertext as follows:

$$\begin{aligned} c_1^* &= g^k g_2^{\omega}, \\ c_2^* &= (u^{\tau^*} v)^k g_2^{\omega\zeta}, \\ K_0 &= e(g,h)^{\alpha k}, \end{aligned}$$

where $\tau^* = H(c_1^*)$. The challenge ciphertext $C^* = (c_1^*, c_2^*)$ is well-formed according to verification equation (1). To verify this, we observe the correctness as follows:

$$\begin{aligned} & e(c_2^*, gg_3) \\ &= e((u^{\tau^*} v)^k g_2^{\omega\zeta}, gg_3) \\ &= e((u^{\tau^*} v)^k g_2^{\omega\zeta}, g) \cdot e((u^{\tau^* v})^k g_2^{\omega\zeta}, g_3) \\ &= e((u^{\tau^*} v)^k g_2^{\omega\zeta}, g) \cdot 1_{\mathbb{G}_{\mathbb{T}}} \\ &= e((u^{\tau^*} v)^k, g) \cdot e(g_2^{\omega\zeta}, g) \\ &= e(u^{\tau^*} v, g)^k, \end{aligned}$$

$$\begin{aligned} & e(c_1^*, (u^{\tau^*} v) g_3') \\ &= e(g^k g_2^{\omega}, (u^{\tau^*} v) g_3') \\ &= e(g^k, (u^{\tau^*} v) g_3') \cdot e(g_2^{\omega}, (u^{\tau^*} v) g_3') \\ &= e(g^k, (u^{\tau^*} v) g_3') \cdot 1_{\mathbb{G}_{\mathbb{T}}} \\ &= e(g^k, (u^{\tau^*} v)) \cdot e(g^k, g_3') \\ &= e(u^{\tau^*} v, g)^k. \end{aligned}$$

So, we have $e(c_2^*, gg_3) = e(c_1^*, (u^{\tau^*} v) g_3')$.

- **Game**$_5$: This is identical to Game$_4$ with one difference of DecapsulationShare query on $(P_i, C = (c_1, c_2))$ in phase 1. $\mathcal{C}$ randomly selects $(\gamma, \iota, \zeta) \in \mathbb{Z}_N^3, (W_3, W_3') \in \mathbb{G}_{p_3}^2$, and answers the DecapsulationShare query of P_i about c_1, c_2 as follows:

$$\begin{aligned} \mu_{i1} &= SK_i (u^{\tau} v)^{\gamma} W_{3i} \cdot g_2^{\iota\zeta}, \\ \mu_{i2} &= g^{\gamma} W_{3i}' \cdot g_2^{\iota}, \end{aligned}$$

where $\tau = H(c_1)$. The partial decapsulation share (μ_{i1}, μ_{i2}) is well-formed according to equation (2). To verify this, we can see the correctness as follows:

$$\begin{aligned} & e(\mu_{i1}, g) \\ &= e(SK_i (u^{\tau} v)^{\gamma} W_{3i} \cdot g_2^{\iota\zeta}, g) \\ &= e(SK_i (u^{\tau} v)^{\gamma}, g) \cdot e(W_{3i} g_2^{\iota\zeta}, g) \\ &= e(SK_i, g) \cdot e((u^{\tau} v)^{\gamma}, g) \\ &= VK_i \cdot e(u^{\tau} v, g)^{\gamma}, \end{aligned}$$

$$\begin{aligned} & VK_i \cdot e(u^{\tau} v, \mu_{i2}) \\ &= VK_i \cdot e(u^{\tau} v, g^{\gamma} W_{3i}' \cdot g_2^{\iota}) \\ &= VK_i \cdot e(u^{\tau} v, g^{\gamma}) \cdot e(u^{\tau} v, W_{3i}' g_2^{\iota}) \\ &= VK_i \cdot e(u^{\tau} v, g)^{\gamma}. \end{aligned}$$

- **Game**$_6$: This is the last game, identical to Game$_5$, but in the challenge phase the challenger $\mathcal{C}$ first performs the **Encapsulate** algorithm to acquire (K, C^*), and selects a random bit $\delta \in \{0,1\}$ and two random session keys K_0, K_1, then sends (C^*, K_δ) to the adversary $\mathcal{A}$. .

First observe that, as desired, the adversary $\mathcal{A}$'s view in Game$_6$ is identical for either choice of $\delta \in \{0,1\}$, but K_δ is never related to C^* in the experiment, so $|\Pr[\delta' = \delta]|$ is exactly 1/2.

Claim 1: Suppose there exists an algorithm $\mathcal{A}$ that can distinguish Game$_1$ from Game$_0$ with advantage ϵ. Then there is a distinguishing algorithm $\mathcal{D}$ with advantage ϵ in finding a collision of the hash function H.

If $\mathcal{A}$ can distinguish Game$_1$ from Game$_0$, $\mathcal{D}$ will find the collision of the hash function H. Because hash function H is collision-resistant, we conclude that this event happens with negligible probability, as desired.

Claim 2: Suppose there exists an algorithm $\mathcal{A}$ that can distinguish Game$_2$ from Game$_1$ with advantage ϵ. Then there is a distinguishing algorithm $\mathcal{D}$ with advantage $\epsilon/2$ in breaking Assumption 1 or Assumption 2.

Proof. If $\mathcal{A}$ can produce $C = (c_1, c_2)$ such that $\tau \neq \tau^* \pmod{N}$ and $\tau = \tau^* \pmod{p_2}$, where $\tau = H(c_1), \tau^* = H(c_1^*)$, $\mathcal{D}$ can find a non-trivial factor of N by computing $\gcd(\tau - \tau^*, N)$. According to Lemma 1, $\mathcal{D}$ can break Assumption 1 or Assumption 2 with advantage $\geq \epsilon/2$ (proof is similar to Lemma 1 of [16]).

Claim 3: Suppose there exists an algorithm $\mathcal{A}$ that can distinguish Game$_3$ from Game$_2$ with advantage ϵ. Then there is a distinguishing algorithm $\mathcal{D}$ with advantage ϵ in breaking Assumption 1.

The only situation is when $\mathcal{A}$ issues a partial decapsulation share extraction oracle with a valid ciphertext (c_1, c_2) such that $c_1 = c_1^*$ and $c_2 \neq c_2^*$. Since $e(g, c_2) = e(gg_3, c_2) = e(c_1, u^{\tau} v g_3') = e(c_1, u^{\tau^*} v g_3') = e(c_1^*, u^{\tau^*} v) = e(g, c_2^*)$, this means that the difference between c_2 and c_2^* is that c_2 has a non-trivial component in $\mathbb{G}_{p_2}$, but c_2^* has no non-trivial component in $\mathbb{G}_{p_2}$ (equation 1 rules out the existence of a G_{p_3} component in c_2). So the relationship between c_2 and c_2^* is that $c_2 = c_2^* \cdot X_1X_2$, where X_1 is an element in G_{p_1} and X_2 is an element in G_{p_2}. Since $e(g, c_2) = e(g, c_2^* \cdot X_1X_2) = e(g, c_2^*)e(g, X_1)e(g, X_2) = e(g, c_2^*)e(g, X_1)$, this mean that $e(g, X_1) = 1_{\mathbb{G}_\mathbb{T}}$, but $g \neq 1_{\mathbb{G}_{p_1}}$, so, we can conclude that $X_1 = 1_{\mathbb{G}_{p_1}}$. $\mathcal{D}$ calculates $\xi = e(T, c_2)/e(T, c_2^*)$. If $\xi = 1_{\mathbb{G}_\mathbb{T}}$, $\mathcal{D}$ can determine $T \in \mathbb{G}_{p_1}$, otherwise $T \in \mathbb{G}_{p_1p_2}$.

Claim 4: Suppose there exists an algorithm $\mathcal{A}$ that can distinguish Game$_4$ from Game$_3$ with advantage ϵ. There is a distinguishing algorithm $\mathcal{D}$ with advantage ϵ in breaking Assumption 1.

Proof. $\mathcal{D}$ begins by taking an instance $(\mathbb{G}, \mathbb{G}_\mathbb{T}, N, e, g, X_3, T)$ of Assumption 1. We now describe how it "interpolates" between Game$_3$ and Game$_4$ with $\mathcal{A}$ using these parameters.

Init. $\mathcal{D}$ first selects a collision-resistance hash function $H : \mathbb{G} \to \mathbb{Z}_N$, four random integers $a, b, c, \alpha \in \mathbb{Z}_N$ and a random polynomial $f(X)$ of degree $t-1$ such that $f(0) = \alpha$. Then it computes $g = g, u = g^a, v = g^b, h = g^c, X_{p_3} = X_3, e(g, h)^\alpha$, and P_i's decryption key share $SK_i = h^{f(i)}Z_{3,i}$ and verification shadow $VK_i = e(g, SK_i)$, for $i = 1, \ldots n$, where $Z_{3,i} \in_R \mathbb{G}_{p_3}$. Finally, $\mathcal{D}$ sends $PK = (\mathbb{G}, \mathbb{G}_\mathbb{T}, N, g, u, v, X_{p_3}, e(\cdot,\cdot), H, e(g, h)^\alpha)$, $VK = (VK_1, \ldots, VK_n)$ to $\mathcal{A}$.

Even though $\mathcal{D}$ knows everyone's decryption key share, it cannot distinguish $T \in \mathbb{G}_{p_1p_2}$ or $T \in \mathbb{G}_{p_1}$.

Phase 1. This phase is identical to phase 1 of Game$_3$.

Challenge. Once the adversary $\mathcal{A}$ ends Phase 1, $\mathcal{D}$ can generate the challenge ciphertext as follows:

$$\begin{aligned} c_1^* &= T, \\ c_2^* &= T^{a\tau^*+b}, \\ K_0 &= e(T, h)^\alpha, \end{aligned}$$

where $\tau^* = H(c_1^*)$. $\mathcal{D}$ now selects a random bit $\delta \in \{0,1\}$ and random session key K_1. $\mathcal{D}$ then sends $(C^* = (c_1^*, c_2^*), K_\delta)$ to $\mathcal{A}$.

Phase 2. This phase is identical to phase 2 of Game$_3$.

Guess. $\mathcal{A}$ outputs a guess bit $\delta' \in \{0,1\}$ for δ.

If $T \in \mathbb{G}_{p_1}$, there exists a number $k \in \mathbb{Z}_N$ to satisfy $T = g^k$, and $\mathcal{D}$ correctly simulates Game$_3$. To verify correctness, notice that we rewrite c_1^*, c_2^* as follows:

$$\begin{aligned} c_1^* &= g^k, \\ c_2^* &= T^{a\tau^*+b} \\ &= (g^{a\tau^*+b})^k \\ &= (u^{\tau^*}v)^k. \end{aligned}$$

If $T \in \mathbb{G}_{p_1p_2}$, there exist two numbers $k, \omega \in \mathbb{Z}_N$ to satisfy $T = g^k g_2^\omega$, and $\mathcal{D}$ correctly simulates the Game$_4$. To verify correctness, observe that we rewrite c_1^*, c_2^* as follows:

$$\begin{aligned} c_1^* &= g^k g_2^\omega, \\ c_2^* &= T^{a\tau^*+b} \\ &= (g^k g_2^\omega)^{a\tau^*+b} \\ &= (g^k)^{a\tau^*+b}(g_2^\omega)^{a\tau^*+b} \\ &= (u^{\tau^*}v)^k(g_2^\omega)^{a\tau^*+b} \\ &= (u^{\tau^*}v)^k g_2^{\omega\zeta}, \end{aligned}$$

where $\zeta = a\tau^* + b$. If $\mathcal{A}$ can distinguish Game$_4$ from Game$_3$ with a advantage ϵ, $\mathcal{D}$ can use the output of $\mathcal{A}$ to distinguish $T \in \mathbb{G}_{p_1}$ or $T \in \mathbb{G}_{p_1p_2}$ with advantage ϵ.

Claim 5: Suppose there exists an algorithm $\mathcal{A}$ that can distinguish Game$_5$ from Game$_4$ with advantage ϵ. Then there is a distinguishing $\mathcal{D}$ with advantage ϵ in breaking Assumption 2.

Proof. $\mathcal{D}$ begins by taking in an instance $(\mathbb{G}, \mathbb{G}_\mathbb{T}, N, e, g, X_1X_2, Z_3, Y_2Y_3, T)$ of the Assumption 2. We now describe how it effectively "interpolates" between Game$_4$ and Game$_5$ with $\mathcal{A}$ using these parameters.

Init. $\mathcal{D}$ first selects a collision-resistance hash function $H : \mathbb{G} \to \mathbb{Z}_N$, four random integers $a, b, c, \alpha \in \mathbb{Z}_N$ and a random polynomial $f(X)$ of degree $t-1$ such that $f(0) = \alpha$. Then it computes $g = g, u = g^a, v = g^b, h = g^c, X_{p_3} = Z_3, e(g, h)^\alpha$, and P_i's decryption key share $SK_i = h^{f(i)}Z_{3,i}$ and verification shadow $VK_i = e(g, SK_i)$, for $i = 1, \ldots n$, where $Z_{3,i} \in_R \mathbb{G}_{p_3}$. Finally, $\mathcal{D}$ sends $PK = (\mathbb{G}, \mathbb{G}_\mathbb{T}, N, g, u, v, X_{p_3}, e(\cdot,\cdot), H, e(g, h)^\alpha)$, $VK = (VK_1, \ldots, VK_n)$ to $\mathcal{A}$.

Phase 1. This phase is identical to Phase 1 of Game 4, with one difference in the DecapsulationShare query in that $\mathcal{D}$ chooses two random $W_3, W_3' \in \mathbb{G}_{p_3}$ and generates the partial decapsulation share of P_i with (c_1, c_2) as follows:

$$\begin{aligned} \mu_{i1} &= SK_i \cdot T^{a\cdot\tau+b} \cdot W_3, \\ \mu_{i2} &= T \cdot W_3', \end{aligned}$$

where $\tau = H(c_1)$.

Challenge. Once the adversary $\mathcal{A}$ ends Phase 1, $\mathcal{D}$ can generate the challenge ciphertext as follows:

$$\begin{aligned} c_1^* &= X_1X_2, \\ c_2^* &= (X_1X_2)^{a\tau^*+b}, \\ K_0 &= e(X_1X_2, h)^\alpha, \end{aligned}$$

where $\tau^* = H(c_1^*)$. $\mathcal{D}$ now selects a random bit $\delta \in \{0,1\}$ and a random session key K_1. $\mathcal{D}$ then sends $(C^* = (c_1^*, c_2^*), K_\delta)$ to $\mathcal{A}$.

There exists $(k, \omega) \in \mathbb{Z}_N^2$ such that $X_1 = g^k$ and $X_2 = g_2^\omega$, and we rewrite (c_1^*, c_2^*) as follows:

$$\begin{aligned} c_1^* &= g^k g_2^\omega, \\ c_2^* &= (g^k g_2^\omega)^{a\tau^*+b} \\ &= (g^k)^{a\tau^*+b}(g_2^\omega)^{a\tau^*+b} \\ &= (u^{\tau^*}v)^k(g_2^\omega)^{a\tau^*+b} \\ &= (u^{\tau^*}v)^k g_2^{\omega\zeta}, \\ K_0 &= e(X_1X_2, h)^\alpha \\ &= e(g^k g_2^\omega, h)^\alpha \\ &= e(g^k, h)^\alpha e(g_2^\omega, h)^\alpha \\ &= e(g, h)^{\alpha k}, \end{aligned}$$

where $\tau^* = H(c_1^*)$ and $\zeta = a\tau^* + b$. So, the form of (c_1^*, c_2^*) is the same as that of Game_4's.

Phase 2. This phase is identical to the phase 2 of Game_4.

Guess. $\mathcal{A}$ outputs a guess bit $\delta' \in \{0,1\}$ for δ.

If $T \in \mathbb{G}_{p_1p_3}$, there exists two numbers $\beta, \iota \in \mathbb{Z}_N$ to satisfy $T = g^\beta g_3^\iota$, and then $\mathcal{D}$ correctly simulates the game_4. To verify correctness, notice that we rewrite μ_{i1}, μ_{i2} as follows:

$$\begin{aligned}\mu_{i1} &= SK_i \cdot T^{a\cdot\tau+b} \cdot W_3\\ &= SK_i \cdot (g^\beta g_3^\iota)^{a\cdot\tau+b} \cdot W_3\\ &= SK_i \cdot (g^\beta)^{a\cdot\tau+b}(g_3^\iota)^{a\cdot\tau+b} \cdot W_3\\ &= SK_i \cdot (u^\tau v)^\beta (g_3^\iota)^{a\cdot\tau+b} \cdot W_3\\ &= SK_i \cdot (u^\tau v)^\beta W_{3i},\\ \mu_{i2} &= T \cdot W_3'\\ &= g^\beta g_3^\iota \cdot W_3'\\ &= g^\beta W_{3i}',\end{aligned}$$

where $W_{3i} = (g_3^\iota)^{a\cdot\tau+b} W_3$ and $W_{3i}' = g_3^\iota \cdot W_3'$.

If $T \notin \mathbb{G}_{p_1p_3}$, but $T \in \mathbb{G}$, so there exists three numbers $\beta, \iota, \sigma \in \mathbb{Z}_N$ to satisfy $T = g^\beta g_2^\iota g_3^\sigma$, and then $\mathcal{D}$ correctly simulates the game_5. To verify correctness, notice that we rewrite μ_{i1}, μ_{i2} as follows:

$$\begin{aligned}\mu_{i1} &= SK_i \cdot T^{a\cdot\tau+b} \cdot W_3\\ &= SK_i \cdot (g^\beta g_2^\iota g_3^\sigma)^{a\cdot\tau+b} \cdot W_3\\ &= SK_i \cdot (g^\beta)^{a\cdot\tau+b}(g_2^\iota)^{a\cdot\tau+b}(g_3^\sigma)^{a\cdot\tau+b} \cdot W_3\\ &= SK_i \cdot (u^\tau v)^\beta (g_2^\iota)^{a\cdot\tau+b}(g_3^\sigma)^{a\cdot\tau+b} \cdot W_3\\ &= SK_i \cdot (u^\tau v)^\beta g_2^{\iota\zeta} W_{3i},\\ \mu_{i2} &= T \cdot W_3'\\ &= g^\beta g_2^\iota g_3^\sigma \cdot W_3'\\ &= g^\beta g_2^\iota W_{3i}',\end{aligned}$$

where $\zeta = a\cdot\tau+b, W_{3i} = (g_3^\sigma)^{a\cdot\tau+b} \cdot W_3, W_{3i}' = g_3^\sigma \cdot W_3'$.

If $\mathcal{A}$ can distinguish Game_5 from Game_4 with advantage ϵ, $\mathcal{D}$ uses the output of $\mathcal{A}$ to decide whether or not $T \in \mathbb{G}_{p_1p_3}$ with advantage ϵ.

Claim 6: Suppose there exists an algorithm $\mathcal{A}$ that can distinguish Game_6 from Game_5 with advantage ϵ. Then there is a distinguishing $\mathcal{D}$ with advantage ϵ in breaking Assumption 3.

$\mathcal{D}$ begins by taking in an instance $(\mathbb{G}, \mathbb{G}_\mathbb{T}, N, e, g, g^\alpha X_2, X_3, g^\beta Y_2, Z_2, T)$ of the Assumption 3. We now describe how it effectively "interpolates" between Game_5 and Game_6 with $\mathcal{A}$ using these parameters.

Init. $\mathcal{D}$ first selects a collision-resistance hash function $H : \mathbb{G} \to \mathbb{Z}_N$, three random integers $a, b, c \in \mathbb{Z}_N$ and a random polynomial $f(X)$ of degree $t-1$ such that $f(0) = 1$. Then it computes $g = g, u = g^a, v = g^b, h = g^c, X_{p_3} = X_3, e(g,h)^\alpha = e(g^\alpha X_2, h)$, and P_i's decryption key share $SK_i = (g^\alpha X_2)^{cf(i)} Z_{3,i}$ and verification shadow $VK_i = e(g, SK_i)$, for $i = 1, \ldots n$, where $Z_{3,i} \in_R \mathbb{G}_{p_3}$. Finally, $\mathcal{D}$ sends $PK = (\mathbb{G}, \mathbb{G}_\mathbb{T}, N, g, u, v, X_{p_3}, e(\cdot,\cdot), H, e(g,h)^\alpha)$, $VK = (VK_1, \ldots, VK_n)$ to $\mathcal{A}$.

Phase 1. This phase is identical to Phase 1 of Game 5, with one difference in the DecapsulationShare query in that $\mathcal{D}$ chooses three random numbers $\gamma, \zeta, \xi \in \mathbb{Z}_N$ and generates the partial decapsulation share of P_i with (c_1, c_2) as follows:

$$\begin{aligned}\mu_{i1} &= SK_i(u^\tau v)^\gamma (Z_2X_3)^\zeta,\\ \mu_{i2} &= g^\gamma \cdot (Z_2X_3^\xi),\end{aligned}$$

where $\tau = H(c_1)$. Suppose that $Z_2 = g_2^\iota, X_3 = g_3^\sigma$, where $(\iota, \sigma) \in \mathbb{Z}_N^2$, to verify the correctness, notice that we rewrite μ_{i1}, μ_{i2} as follows:

$$\begin{aligned}\mu_{i1} &= SK_i(u^\tau v)^\gamma (Z_2X_3)^\zeta\\ &= SK_i(u^\tau v)^\gamma g_2^{\iota\zeta} \cdot g_3^{\sigma\zeta}\\ &= SK_i(u^\tau v)^\gamma \cdot g_2^{\iota\zeta} \cdot V_3,\\ \mu_{i2} &= g^\gamma \cdot (Z_2X_3^\xi)\\ &= g^\gamma \cdot g_2^\iota \cdot g_3^{\sigma\xi}\\ &= g^\gamma \cdot g_2^\iota \cdot V_3',\end{aligned}$$

where $V_3 = g_3^{\sigma\zeta}$ and $V_3' = g_3^{\sigma\xi}$. The form of μ_{i1}, μ_{i2} is the same as with Game_5's.

Challenge. Once the adversary $\mathcal{A}$ ends phase 1, $\mathcal{D}$ can generate the challenge ciphertext as follows:

$$\begin{aligned}c_1^* &= g^\beta Y_2,\\ c_2^* &= (g^\beta Y_2)^{a\tau^*+b},\\ K_0 &= T^c,\end{aligned}$$

where $\tau^* = H(c_1^*)$. $\mathcal{D}$ now selects a random bit $\delta \in \{0,1\}$ and random session key K_1. $\mathcal{D}$ then sends $(C^* = (c_1^*, c_2^*), K_\delta)$ to $\mathcal{A}$.

To verify the correctness of (c_1^*, c_2^*), we assume that $Y_2 = g_2^\omega$, where $\omega \in \mathbb{Z}_N$, and rewrite (c_1^*, c_2^*) as follows:

$$\begin{aligned}c_1^* &= g^\beta Y_2\\ &= g^\beta g_2^\omega,\\ c_2^* &= (g^\beta Y_2)^{a\tau^*+b}\\ &= (g^\beta)^{a\tau^*+b}(Y_2)^{a\tau^*+b}\\ &= (u^{\tau^*}v)^\beta (Y_2)^{a\tau^*+b}\\ &= (u^{\tau^*}v)^\beta (g_2^\omega)^{a\tau^*+b}\\ &= (u^{\tau^*}v)^\beta g_2^{\omega\zeta},\end{aligned}$$

where $\tau^* = H(c_1^*)$ and $\zeta = a\tau^* + b$.

Phase 2. This phase is identical phase 2 of Game_5.

Guess. $\mathcal{A}$ outputs a guess bit $\delta' \in \{0,1\}$ for δ.

If $T = e(g,g)^{\alpha\beta}$, $\mathcal{D}$ correctly simulates the game_5, since $K_0 = T^c = e(g,g)^{\alpha\beta c} = e(g, g^c)^{\alpha\beta} = e(g,h)^{\alpha\beta}$.

If $T \in_R \mathbb{G}_\mathbb{T}$, $K_0 = T^c$ is a random value of $\mathbb{G}_\mathbb{T}$. In this case, the challenge ciphertext (c_1^*, c_2^*) carries no information on K_0 or K_1.

If $\mathcal{A}$ can distinguish Game_6 from Game_5 with a advantage ϵ, $\mathcal{D}$ uses the output of $\mathcal{A}$ to decide whether or not $T = e(g,g)^{\alpha\beta}$ with advantage ϵ. ■

VI. Comparison

In this section, we compare our TKEM scheme with the BMW'05 TKEM and IAHS'07 TKEM schemes. The BMW'05 scheme is the provable security under the DBD-H assumption in the standard model with loose reduction. The IAHS'07 TKEM scheme is the provable security under the RSA assumption in the random oracle model with tight reduction. Neither BMW'05 nor IAHS'07 can withstand an adaptive corruption attack. Here, SM is the abbreviation of the standard model and ROM is the abbreviation of the random oracle model. In Table 1 we summarize the comparisons.

TABLE I.
Comparisons with other TKEM schemes

	Adaptive corruption	Secure level	Secure model	Security reduction
BMW'05	×	IND-CCA	SM	loose
IAHS'07	×	IND-CCA	ROM	tight
OURS	√	IND-CCA	SM	tight

VII. Conclusions

In this paper, we have provided a threshold KEM scheme against chosen ciphertext attacks and with constant size ciphertext that is fully secure in the standard model from some assumptions about bilinear groups with composite order. In doing so, we discovered that the static-corruption-secure Boyen-Mei-Waters KEM can be proven to be adaptive-corruption-secure if we use bilinear groups with composite order and the dual system approach of Lewko-Waters.

Acknowledgment

The authors are grateful to the anonymous referees for their valuable comments and suggestions to improve the presentation of this paper.

References

[1] R. Cramer and V. Shoup, "A practical public key cryptosystem provably secure against adaptive chosen ciphertext attack," in *Advances in Cryptology ł CRYPTO '98*, ser. Lecture Notes in Computer Science, H. Krawczyk, Ed. Springer Berlin, 1998, vol. 1462, pp. 13–25.

[2] V. Shoup and R. Gennaro, "Securing threshold cryptosystems against chosen ciphertext attack," in *Advances in Cryptology - EUROCRYPT'98*, ser. Lecture Notes in Computer Science, K. Nyberg, Ed. Springer Berlin, 1998, vol. 1403, pp. 1–16.

[3] J. Qin and H. Zhao, "k out of n oblivious transfer protocols from bilinear pairings," *Journal of Software*, vol. 5, no. 1, pp. 65–72, 2010.

[4] D. Boneh, X. Boyen, and S. Halevi, "Chosen ciphertext secure public key threshold encryption without random oracles," in *Topics in Cryptology C CT-RSA 2006*, ser. Lecture Notes in Computer Science, D. Pointcheval, Ed. Springer Berlin, 2006, vol. 3860, pp. 226–243.

[5] R. Canetti, I. Damgaard, S. Dziembowski, Y. Ishai, and T. Malkin, "On adaptive vs. non-adaptive security of multiparty protocols," in *Advances in Cryptology - EUROCRYPT 2001*, ser. Lecture Notes in Computer Science, B. Pfitzmann, Ed. Springer Berlin, 2001, vol. 2045, pp. 262–279.

[6] S. Jarecki and A. Lysyanskaya, "Adaptively secure threshold cryptography: Introducing concurrency, removing erasures," in *Advances in Cryptology - EUROCRYPT'2000*, ser. Lecture Notes in Computer Science, B. Preneel, Ed. Springer Berlin, 2000, vol. 1807, pp. 221–242.

[7] C. Gentry and B. Waters, "Adaptive security in broadcast encryption systems (with short ciphertexts)," in *Advances in Cryptology - EUROCRYPT 2009*, ser. Lecture Notes in Computer Science, A. Joux, Ed. Springer Berlin, 2009, vol. 5479, pp. 171–188.

[8] M. Abe and S. Fehr, "Adaptively secure feldman vss and applications to universally-composable threshold cryptography," in *Advances in Cryptology C CRYPTO 2004*, ser. Lecture Notes in Computer Science, M. Franklin, Ed. Springer Berlin, 2004, vol. 3152, pp. 317–334.

[9] M. Abe, R. Gennaro, K. Kurosawa, and V. Shoup, "Tag-kem/dem: A new framework for hybrid encryption and a new analysis of kurosawa-desmedt kem," in *Advances in Cryptology C EUROCRYPT 2005*, ser. Lecture Notes in Computer Science, R. Cramer, Ed. Springer Berlin, 2005, vol. 3494, pp. 618–634.

[10] X. Boyen, Q. Mei, and B. Waters, "Direct chosen ciphertext security from identity-based techniques," in *Proceedings of the 12th ACM conference on Computer and communications security*, ser. Lecture Notes in Computer Science. Alexandria, VA, USA: ACM, 2005, pp. 320–329.

[11] E. Kiltz, "Chosen-ciphertext secure key-encapsulation based on gap hashed diffie-hellman," in *Public Key Cryptography- PKC 2007*, ser. Lecture Notes in Computer Science, T. Okamoto and X. Wang, Eds. Springer Berlin, 2007, vol. 4450, pp. 282–297.

[12] T. Ishihara, H. Aono, S. Hongo, and J. Shikata, "Construction of threshold (hybrid) encryption in the random oracle model: How to construct secure threshold tag-kem from weakly secure threshold kem," in *Information Security and Privacy*, ser. Lecture Notes in Computer Science, J. Pieprzyk, H. Ghodosi, and E. Dawson, Eds. Springer Berlin, 2007, vol. 4586, pp. 259–273.

[13] Y. Ming, X. Shen, and Y. Peng, "Provably security identity-based sanitizable signature scheme without random oracles," *Journal of Software*, vol. 6, no. 10, pp. 1890–1897, 2011.

[14] C. Delerablée and D. Pointcheval, "Dynamic threshold public-key encryption," in *Advances in Cryptology C CRYPTO 2008*, ser. Lecture Notes in Computer Science, D. Wagner, Ed. Springer Berlin, 2008, vol. 5157, pp. 317–334.

[15] B. Libert and M. Yung, "Adaptively secure non-interactive threshold cryptosystems," in *The 38th International Colloquium on Automata, Languages and Programming (ICALP 2011)*, ser. Lecture Notes in Computer Science, L. Aceto, M. Henzinger, and J. Sgall, Eds., 2011, vol. 6756, pp. 588–600.

[16] A. Lewko and B. Waters, "New techniques for dual system encryption and fully secure hibe with short ciphertexts," in *Theory of Cryptography*, ser. Lecture Notes in Computer Science, D. Micciancio, Ed. Springer Berlin, 2010, vol. 5978, pp. 455–479.

[17] V. Shoup, "Sequences of games: a tool for taming complexity in security proofs," Cryptology ePrint Archive, Report 2004/332, 2004, http://eprint.iacr.org/cgi-bin/cite.pl?entry=2004/332.

Yuanju Gan 1973- received a B.S. in communication engineering from Changsha Railway Institute in China in 1996, and M.S. in computer applications from Central South University in China in 2003. He is currently a doctoral student at Beijing University of Posts and Telecommunications. His research interests include cryptography and information security.

Licheng Wang 1972- received the B.S. degree in Computer Science from Northwest Normal University, China, in 1995, and the M.S. degree in mathematics from Nanjing University, China, in 2001, and the Ph.D. degree in Computer Science from Shanghai Jiao Tong University, China, in 2007, respectively. He is currently an assistant professor at Beijing University of Posts and Telecommunications. His current research interests include cryptography, information security

Jianhua Yan 1977- received a B.S. in chemistry engineering from JiLin university in China in 2002, and M.S. in computer applications from Liaoning shihua university in China in 2005. He is currently a doctoral student at Beijing University of Posts and Telecommunications. His research interests include cryptography and information security.

Yixian Yang 1961-, received the B.S. degree in applied mathematics from Chengdu Institute of Telecommunication Engineering, China, in 1983, and M.S. degree in applied mathematics and Ph.D. degree in signal and information processing from Beijing University of Posts and Telecommunications(BUPT), China, in 1986 and 1988, respectively. He is a professor of BUPT from 1992. His research interests are information security, network security, coding, cryptography, quantum cipher, chaos, and fuzzy systems.