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

Yuanju Gan\textsuperscript{a,b}, Licheng Wang\textsuperscript{a}, Jianhua Yan\textsuperscript{a}, Yixian Yang\textsuperscript{a}
\textsuperscript{a} Information Security Center, Beijing University of Posts and Telecommunications, Beijing 100876, China
Email: \{ganyj66, wanglec, yjh, yxyang\}@bupt.edu.cn
\textsuperscript{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 (CCA) 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-
Anisms [10], [12], [14] have been studied in a static corruption model, that is, an adversary chooses which users it wants to corrupt before the scheme is setup. However, in adaptive corruption model, the 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 $\mathbb{G}_T$ be two cyclic groups of order $N = p_1 p_2 p_3$ (where $p_1, p_2, p_3$ are distinct primes). A bilinear map $e(\cdot, \cdot)$ is a map $\mathbb{G} \times \mathbb{G} \rightarrow \mathbb{G}_T$ 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 $e(g, h) = 1_{\mathbb{G}_T}$, 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 $\mathbb{G}_T$, for any $h_i \in G_{p_i}$ and $h_j \in G_{p_j}$ ($i \neq j$), that is $e(h_i, h_j) = 1_{\mathbb{G}_T}(i \neq j)$.

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 $(N = p_1 p_2 p_3, \mathbb{G}, \mathbb{G}_T, e(\cdot, \cdot))$, as well as $g \in \mathbb{G} G_{p_1} X_1 \in \mathbb{G} G_{p_2}$ and $T \in \mathbb{G}$, it is infeasible to efficiently decide whether $T \in \mathbb{G}_{p_1 p_2}$ or $T \in \mathbb{G}_{p_2}$.

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

**Assumption 3** ([16]): A description of $(N = p_1 p_2 p_3, \mathbb{G}, \mathbb{G}_T, e(\cdot, \cdot))$, a set of group elements $(g, g^\alpha X_2, X_3, g^\beta Y_2, Z_2)$ and $T \in \mathbb{G}_T$, where $g \in \mathbb{G} G_{p_1}, (X_2, Y_2, Z_2) \in \mathbb{G}_{p_1}^3$, $X_1 \in \mathbb{G}_{p_2}$ and $(\alpha, \beta) \in \mathbb{Z}_N^2$, it is infeasible to efficiently decide whether $T = e(g, g)^{\alpha \beta}$.

**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($\Lambda, t, n$):** Takes as input a security parameter $\Lambda$, decryption threshold $t$, and a number of decryption participants $n$. It outputs a set of parameters $(PK, SK, VK)$, where $PK$ is the public key, $SK = (SK_1, \cdots, SK_n)$ and $VK = (VK_1, \cdots, VK_n)$ are the corresponding decryption keys and verification keys, respectively. The $i$th 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 $\mu_i$ of the ciphertext $C$, or a special symbol $(\bot, \bot)$.
- **ShareVerify(PK, VK_i, C, \mu_i):** Takes as input the public key $PK$, verification keys $VK_i$, as well as a ciphertext $C$ and partial decapsulation share $\mu_i$. It checks whether $\mu_i$ is a valid partial decapsulation share with respect to $VK_i$. It outputs valid or invalid.
- **Reconstruct(PK, VK, C, \Omega):** Takes as input the public key $PK$, verification keys $VK$, 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 $\bot$.

Let $(PK, SK, VK)$ be the output of the $\text{Setup}(n, t, \lambda)$. We require the following two consistency properties:

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

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 $\lambda$.

- **Init:** The challenger $\mathcal{C}$ runs Setup($n, t, \lambda$) algorithm to obtain the set of parameters $PK, SK = (SK_1, \ldots, SK_n)$, and $VK = (VK_1, \ldots, VK_n)$. It gives $PK, VK$ 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 $i$th 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.

  **Decapsulation/Share query:** The adversary $\mathcal{A}$ adaptively issues Decapsulation/Share query with $(PK, C)$, where $i \in \{1, \ldots, n\}$. The challenger $\mathcal{C}$ runs the PartialDecapsulation 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 $(K_0, C^*)$ 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 Decapsulation/Share 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 $Adv_{\mathcal{A}, n, t}$ 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 $0 < t \leq n$, and the advantage of any probabilistic polynomial-time (PPT) adversary $\mathcal{A}$, the advantage $Adv_{\mathcal{A}, n, t}^{IND-ACA-TCCA}$ is $|\text{Pr}[\delta' = \delta] - 1/2|$ is negligible with $\lambda$.

### III. The Proposed Scheme

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

- **Setup($n, t, \lambda$).** Given the parameter $\lambda, t, n$, this algorithm does the following:
  1) Select bilinear groups $(\mathbb{G}, \mathbb{G}_T)$ 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 $e : \mathbb{G} \times \mathbb{G} \to \mathbb{G}_T$;
  2) Select generators $(g, h, u, v) \in \mathbb{G}^4$, $X_p, Z_p \in \mathbb{G}_p$;

- **Encapsulate($PK$).** Given the $PK$, this algorithm first picks $k \in \mathbb{Z}_N$ and computes
  \[ c_1 = g^k, \quad c_2 = (u^\tau v)^k, \]
  where $\tau = H(c_1)$. The complete encapsulated key, $C$ is the two group elements $(c_1, c_2)$. The session key, $K = e(g, h)^{sk}$, 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 $g_3, g_3' \in \mathbb{G}_p$, and checks whether or not
  \[ e(c_2, g_3) \neq e(c_1, (u^\tau v)g_3'), \]  (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 $\beta_i \in \mathbb{Z}_N$, $(W_{3i}, W_{3i}') \in \mathbb{G}_p^2$, and computes
  \[ \mu_1 = SK_i \cdot (u^\tau v)^{\beta_i} \cdot W_{3i}, \]
  \[ \mu_2 = g^\mu W_{3i}', \]
  where $\tau = H(c_1)$. Then participant $P_i$ can send $\mu_i \in \{\mu_1, \mu_2\}$ to the combiner through secure channels.

- **ShareVerify($PK, VK, C, \mu_i$).** The combiner verifies whether $\mu_i = \{\mu_1, \mu_2\}$ generated by the $i$th participant is valid, as follows. It first computes $\tau = H(c_1)$, then checks whether or not
  \[ e(\mu_1, g) = VK_i \cdot e(u^\tau v, \mu_2), \]  (2)
  is correct. If so, the algorithm outputs valid. Otherwise it outputs invalid.

- **Reconstruct($PK, VK, C, \Omega$).** 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
\[
\begin{align*}
  d_1 &= \prod_{i=1}^{t} \mu_{1i}^\lambda, \\
  d_2 &= \prod_{i=1}^{t} \mu_{2i}^\lambda,
\end{align*}
\]
where $\lambda_i = \prod_{j=1,j \neq i}^{t} \lambda_j$. Finally, the combiner uses $(c_1, c_2, d_1, d_2)$ to reconstruct session key $K$ as follows:
\[
K = \frac{e(c_1, d_1)}{e(c_2, d_2)}.
\]

IV. SECURITY ANALYSIS

A. Correctness
The consistency of equation (1) is given by
\[
e(c_2, g_{33}) = e((u^r v)^k, g_{33}) = e((u^r v)^k, g_1 e((u^r v)^k, g_3)),
\]
e($c_1$, ($u^r v$)$g_{33}$) = $e(g_k, (u^r v)g_1^t)$ = $e(g_k, (u^r v)g_1^t)$,
and $e(u^r v, g_3) = e(g_k, g_3) = 1$.
The consistency of the equation (2) is given by
\[
e(\mu, g) = e(SK_i, (u^r v)^{\beta_i} \cdot W_{3i}, g)
\]
\[
e(SK_i, g) \cdot e((u^r v)^{\beta_i}, g) \cdot e(W_{3i}, g)
\]
\[
= VK_i \cdot e((u^r v, g_{3i}^t)
\]
\[
= VK_i \cdot e(u^r v, g_{3i}^t)
\]
\[
= VK_i \cdot e(u^r v, g_{3i}^t).
\]
The consistency of the equation (3) is given by
\[
\begin{align*}
  d_1 &= \prod_{i=1}^{t} \mu_{1i}^\lambda \\
  &= \prod_{i=1}^{t} (h^{f(i)} Z_{3i}(u^r v)^{\beta_i} \cdot W_{3i})^\lambda
\end{align*}
\]
\[
\begin{align*}
  &= \prod_{i=1}^{t} (h^{f(i)} (u^r v)^{\beta_i})^\lambda \prod_{i=1}^{t} (Z_{3i}, W_{3i})^\lambda
\end{align*}
\]
\[
\begin{align*}
  &= \prod_{i=1}^{t} h^{f(i)\lambda} \prod_{i=1}^{t} (u^r v)^{\beta_i} \cdot R_{3i}^t
\end{align*}
\]
\[
= h^\alpha \prod_{i=1}^{t} (u^r v)^{\beta_i} \cdot R_{3i}^t,
\]
and
\[
\begin{align*}
  d_2 &= \prod_{i=1}^{t} \mu_{2i}^\lambda \\
  &= \prod_{i=1}^{t} (g_k W_{3i}^t)^{\lambda_i} = \prod_{i=1}^{t} (g_k)^{\beta_i} \cdot R_{3i}^t.
\end{align*}
\]

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, Game0, is basically identical to the IND-ACA-TCCA game in the standard model, and the adversary $A$‘s advantage is defined accordingly. In the last game, Game0, the adversary will still have to guess a given bit. But in this last game, the challenger $C$ 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 $(C^*, K_\delta)$ to the adversary $A$. 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, must be very similar to Game$_{i-1}$, that is, the advantage of $A$ in Game, 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 $G_{p_2}$ and $G_{p_3}$, respectively.

• Game1: We now define a game, Game0, that is an interactive computation between a challenger $C$ and adversary $A$. This game is simply the usual IND-ACA-TCCA game, in which $C$ provides the adversary’s environment.

– Initial: $C$ runs the Setup algorithm to obtain the description of $PK = (G, G_T, N, g, u, v, X_p, c(\cdot), h, e(g, h)^\alpha, MSK = (h_1, h_2, h_3, p_1, p_2, p_3)$, and picks a polynomial $f(x) = \alpha + \alpha_1 x + \ldots + \alpha_{t-1} x^{t-1}$ to compute $P_1$‘s decryption key share $SK_i = h^{f(i)} Z_{3i}$ and verification key $VK_i = e(g, SK_i)$, for $i = 1, \ldots, n$, where $Z_{3i} \in G_{p_3}, C$ gives the public key $PK$ and verification key $VK = \{VK_1, \ldots, VK_n\}$ to $A$.

– Phase 1: $A$ can adaptively issue a “Corruption query” and “DecapsulationShare query.”

Corruption query: If $A$ wants to corrupt $P_1$, $C$ just gives the decryption key share $SK_i$ to $A$. No more than $t - 1$ decryption key shares can be obtained by $A$ in the game1.

DecapsulationShare query: Received $(P_1, C = (c_1, c_2))$ from $A$, the challenger $C$ checks whether $C$ is well-formed according to the equality (1). If so, $C$ randomly picks $\beta_i \in Z_{n*}, (W_{3i}, W_{3i}^t) \in G_{p_3}^2$, computes $\mu_{1i} = SK_i \cdot (u^r v)^{\beta_i} \cdot W_{3i}$, $\mu_{2i} = g_{3i}^t W_{3i}$, where $\tau = H(c_1)$, and sends the partial decapsulation share $\mu_i = (\mu_{1i}, \mu_{2i})$ to $A$. Otherwise, $C$ gives a random value to $A$.

– Challenge: Once $A$ ends the Phase 1, $C$ can form the following challenge information, $C$ randomly selects $k \in Z_{n*}$, and computes $C^* = (c_1^* = g_k^t, c_2^* = (u^r v)^k)$, $K_0 = e(g, h)^{\alpha k},$ where $\tau = H(c_1)$, With this, $C$ now selects a random
bit \( \delta \in \{0, 1\} \) and a random session key \( K_1 \). \( C \) then sends \((C^*, K_1)\) to \( A \).

- **Phase 2**: \( A \) 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, \( A \) outputs a guess bit \( \delta' \in \{0, 1\} \) for \( \delta \). Since Game_0 is identical to the IND-ACA-TCCA game, we have

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

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

- **Game**: 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**: This is identical to Game_2, except the challenger will refuse all decryption queries that \( H(c_1) \neq H(c_1^*) \) and \( H(c_1) = H(c_1^*) \) (mod \( p_2 \)).

- **Game**: This is the same as Game_3, except that in the phase 2, \( C \) will abort if \( A \) manages to make PartialDecapsulate query \((F_1, C = (c_1, c_2))\) such that \( C = (c_1, c_2) \) can pass the CipherTextVerify algorithm and for which \( c_1 = c_1^* \) and \( c_2 \neq c_2^* \).

- **Game**: This is the same as Game_3, with one difference in the Challenge phase that \( C \) randomly selects \((k, \omega, \zeta) \in \mathbb{Z}_N^3\) and generates the challenge ciphertext as follows:

\[
ce_1^* = g^{g_2}\sigma_2^* \\\nc_2^* = (u^{-1}v)^{k}\omega g_2^\zeta \\\nK_0 = e(g, h)^{\delta k}
\]

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:

\[
e(c_2^*, g_g) = e(u^{-1}v)^k g_2^{\omega g_2^\zeta} \cdot g_3 \\\ne((u^{-1}v)^k g_2^{\omega g_2^\zeta}) = e((u^{-1}v)^k g_2^{\omega g_2^\zeta}) \cdot 1_{g_2} \\\ne((u^{-1}v)^k g_2^{\omega g_2^\zeta}) \cdot g_2^{g^\zeta} \\\ne(u^{-1}v, g)^k \\\ne(c_1^*, (u^{-1}v)g_2^k) \\\ne(g^{g_2^k}(u^{-1}v)g_2^k) \\\ne(g^{g_2^k}(u^{-1}v)g_2^k) \cdot e(g_2^k, (u^{-1}v)g_2^k) \\\ne(g^{g_2^k}(u^{-1}v)g_2^k) \cdot 1_{g_2} \\\ne(g^{g_2^k}(u^{-1}v)g_2^k) \cdot e(g, g_2^k) \\\ne(u^{-1}v, g)^k.
\]

So, we have \( e(c_2^*, g_g) = e(c_1^*, (u^{-1}v)g_2)^k \).

- **Game**: This is identical to Game_1 with one difference of DecapsulationShare query on \((F_1, C = (c_1, c_2))\) in phase 1. \( C \) randomly selects \((\gamma, \iota, \zeta) \in \mathbb{Z}_N^3\), \((W_3, W_4^\gamma) \in G_2\), and answers the DecapsulationShare query of \( F_1 \) about \( c_1, c_2 \) as follows:

\[
\mu_{i, 1} = SK_i(u^{-1}v)^\gamma W_{3i} \cdot g_2^\zeta \\\n\mu_{i, 2} = g^\gamma W_{3i} \cdot g_2^\zeta,
\]

where \( \tau = H(c_1) \). The partial decapsulation share \((\mu_{i, 1}, \mu_{i, 2})\) is well-formed according to equation (2). To verify this, we can see the correctness as follows:

\[
e(\mu_{i, 1}, g) = e(SK_i(u^{-1}v)^\gamma W_{3i} \cdot g_2^\zeta, g) = e(SK_i(u^{-1}v), g) \cdot e(W_{3i} g_2^\zeta, g) = e(SK_i, g) \cdot e(u^{-1}v, g)^\gamma = VK_i \cdot e(u^{-1}v, g)^\gamma,
\]

\[
VK_i \cdot e(u^{-1}v, g_{i, 3}) = VK_i \cdot e(u^{-1}v, g_{i, 3} W_{3i}^\gamma) = VK_i \cdot e(u^{-1}v, g_{i, 3}) = VK_i \cdot e(u^{-1}v, g_{i, 3})^\gamma.
\]

- **Game**: This is the last game, identical to Game_5, but in the challenge phase the challenger \( C \) 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 \((C^*, K_0)\) to the adversary \( A \).

First observe that, as desired, the adversary \( A \)‘s view in Game_5 is identical for either choice of \( \delta \in \{0, 1\} \), but \( \delta \) is never related to \( C^* \) in the experiment, so \( |Pr[\delta' = \delta] - 1/2| \) is exactly 1/2.

**Claim 1**: Suppose there exists an algorithm \( A \) that can distinguish Game_1 from Game_0 with advantage \( \epsilon \). Then there is a distinguishing algorithm \( D \) with advantage \( \epsilon \) in finding a collision of the hash function \( H \).

If \( A \) can distinguish Game_1 from Game_0, \( D \) 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 \( A \) that can distinguish Game_2 from Game_1 with advantage \( \epsilon \). Then there is a distinguishing algorithm \( D \) with advantage \( \epsilon/2 \) in breaking Assumption 1 or Assumption 2.

If \( A \) can produce \( C = (c_1, c_2) \) such that \( \tau \neq \tau^* \) (mod \( N \)) and \( \tau = \tau^* \) (mod \( p_2 \)), where \( H(c_1), \tau^* = H(c_1^*) \), \( D \) can find a non-trivial factor of \( N \) by computing \( \gcd(\tau^* - \tau, N) \). According to Lemma 1, \( D \) 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 \( A \) that can distinguish Game_3 from Game_2 with advantage \( \epsilon \). Then there is a distinguishing algorithm \( D \) with advantage \( \epsilon \) in breaking Assumption 1.

The only situation is when \( A \) 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 \( e(g, c_2) = e(g_{g_2}, c_2) = e(c_1, u^{-1}v g_2^k) = e(c_1, u^{-1}v g_2^k) = e(c_1, u^{-1}v) \), this means that the difference
Proof. \( H \) sends \( PK \) generate the challenge ciphertext as follows:

\[
\begin{align*}
\alpha &= \frac{1}{\epsilon}
\end{align*}
\]

where \( \xi = a \tau^* + b \). If \( A \) can distinguish Game\(_4\) from Game\(_3\) with a advantage \( \epsilon \), \( D \) can use the output of \( A \) to distinguish \( T \in \mathbb{G}_{p_2} \) or \( T \notin \mathbb{G}_{p_2} \) with advantage \( \epsilon \).

**Claim 5:** Suppose there exists an algorithm \( A \) that can distinguish Game\(_3\) from Game\(_2\) with advantage \( \epsilon \). Then there is a distinguishing \( D \) with advantage \( \epsilon \) in breaking Assumption 2.

Proof. \( D \) begins by taking an instance \((G, G_T, N, e, g, X_1, X_2, Z_3, Y_2 Y_3, T)\) of the Assumption 2. We now describe how it "interpolates" between Game\(_3\) and Game\(_4\) with \( A \) using these parameters.

**Init.** \( D \) first selects a collision-resistance hash function \( H : G \to Z_N \), four random integers \( a, b, c, \alpha \in Z_N \) and a random polynomial \( f(X) \) of degree \( t - 1 \) such that \( f(0) = \alpha \). Then it computes \( g = g, u = g^a, v = g^b, h = g^c, X_p = X, e(g, h)^{\alpha}, \) 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 \( Z_{3,i} \in \mathbb{G}_{p_2} \). Finally, \( D \) sends \( PK = (G, G_T, N, g, u, v, X_p, e(\cdot, \cdot), H, e(g, h)^{\alpha}), VK = (VK_1, \ldots, VK_n) \) to \( A \).

Even though \( D \) knows everyone's decryption key share, it cannot distinguish \( T \in \mathbb{G}_{p_2} \) or \( T \notin \mathbb{G}_{p_2} \).

**Phase 1.** This phase is identical to Phase 1 of Game\(_3\).

**Challenge.** Once the adversary \( A \) begins Phase 1, \( D \) can generate the challenge ciphertext as follows:

\[
\begin{align*}
c_1^i &= T, \\
c_2^i &= T^a \tau^* + b, \\
K_0 &= e(T, h)^{\alpha_a},
\end{align*}
\]

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

**Phase 2.** This phase is identical to Phase 2 of Game\(_3\).

**Guess.** \( A \) outputs a guess bit \( \delta' \in \{0, 1\} \) for \( \delta \).

If \( T \in \mathbb{G}_{p_2} \), there exists a number \( k \in Z_N \) to satisfy \( T = g^k \), and \( D \) correctly simulates Game\(_3\). To verify correctness, notice that we rewrite \( c_1, c_2 \) as follows:

\[
\begin{align*}
c_1^i &= g^k, \\
c_2^i &= T^a \tau^* + b = (g^{a^2}T^a)^{\tau^*} + b = (u^a v)^{\tau^*}.
\end{align*}
\]

If \( T \notin \mathbb{G}_{p_2} \), there exist two numbers \( k, \omega \in Z_N \) to satisfy \( T = g^k g^\omega \), and \( D \) correctly simulates the Game\(_3\). To verify correctness, observe that we rewrite \( c_1^i, c_2^i \) as follows:

\[
\begin{align*}
c_1^i &= g^k g^\omega, \\
c_2^i &= T^a \tau^* + b = (g^{a^2}T^a)^{\tau^*} + b = (u^a v)^{\tau^*}.
\end{align*}
\]

\( \mu_{11} = SK_1, T^a \tau^* + b, W_3, \)

\( \mu_{12} = T, W_3, \)

where \( \tau = H(c_1^i) \).

**Challenge.** Once the adversary \( A \) ends Phase 1, \( D \) can generate the challenge ciphertext as follows:

\[
\begin{align*}
c_1^i &= X_1 X_2, \\
c_2^i &= (X_1 X_2)^{a^2} \tau^* + b, \\
K_0 &= e(X_1 X_2, h)^{\alpha},
\end{align*}
\]

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

There exists \( (k, \omega) \in Z_N^2 \) such that \( X_1 = g^k \) and \( X_2 = g^\omega \), and we rewrite \( c_1^i, c_2^i \) as follows:

\[
\begin{align*}
c_1^i &= g^k g^\omega, \\
c_2^i &= (g^{a^2}T^a)^{\tau^*} + b = (u^a v)^{\tau^*}.
\end{align*}
\]

\( \mu_{11} = e(g^k g^\omega, h)^{\alpha}, \)

\( \mu_{12} = e(g^k h)^{\alpha} e(g^\omega, h)^{\alpha}, \)

\( \mu_{13} = e(g, h)^{\alpha k}, \)
where \( \tau^* = H(c_i^*) \) and \( \zeta = \alpha \tau^* + b \). 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.** A outputs a guess bit \( \delta \in \{0, 1\} \) for \( \delta \).

If \( T \in \mathcal{G}_{p_1p_3} \), there exists two numbers \( \beta, \ell \in \mathbb{Z}_N \) to satisfy \( T = g^{\beta_3}g_3^{\ell} \), and then \( D \) correctly simulates the game_4. To verify correctness, notice that we rewrite \( \mu_{11}, \mu_{12} \) as follows:

\[
\mu_{11} = SK_1 \cdot T^{a \tau^*} \cdot W_3 = SK_1 \cdot (g^b g_3^a)^{a \tau^*} \cdot W_3 = SK_1 \cdot (g^b g_3^a)^{a \tau^*} (g_3^a)^{a \tau^*} \cdot W_3,
\]
\[
\mu_{12} = T \cdot W_3 = g^{a} g_3^{b} \cdot W_3 = g^{a} g_3^{b} W_{31},
\]
where \( W_{31} = (g_3^a)^{a \tau^* + b} \cdot W_3 \).

If \( T \notin \mathcal{G}_{p_1p_3} \), but \( T \in \mathcal{G} \), so there exists three numbers \( \beta, c, \sigma \in \mathbb{Z}_N \) to satisfy \( T = g^{\beta_3}g_3^{c}g_3^{\sigma} \), and then \( D \) correctly simulates the game_3. To verify correctness, notice that we rewrite \( \mu_{11}, \mu_{12} \) as follows:

\[
\mu_{11} = SK_1 \cdot T^{a \tau^*} \cdot W_3 = SK_1 \cdot (g^{b_3} g_3^{a})^{a \tau^* + b} \cdot W_3 = SK_1 \cdot (g^{b_3} g_3^{a})^{a \tau^* + b} (g^{b_3} g_3^{a})^{a \tau^* + b} \cdot W_3,
\]
\[
\mu_{12} = T \cdot W_3 = g^{b_3} g_3^{b} \cdot W_3 = g^{b_3} g_3^{b} W_{31},
\]
where \( W_{31} = (g_3^a)^{a \tau^* + b} \cdot W_3 \).

If \( A \) can distinguish Game_3 from Game_4 with advantage \( \epsilon \), \( D \) uses the output of \( A \) to decide whether or not \( T \in \mathcal{G}_{p_1p_3} \) with advantage \( \epsilon \).

**Claim 6:** Suppose there exists an algorithm \( A \) that can distinguish Game_3 from Game_5 with advantage \( \epsilon \). Then there is a distinguishing \( D \) with advantage \( \epsilon \) in breaking Assumption 3.

\( D \) begins by taking in an instance \((\mathcal{G}, \mathcal{G}_T, N, e, g, g^a X_2, X_3, g^a Y_2, Z_2, T)\) of the Assumption 3. We now describe how it effectively “interpolates” between Game_5 and Game_4 with \( A \) using these parameters.

**Init.** \( D \) first selects a collision-resistance hash function \( H : \mathcal{G} \rightarrow \mathbb{Z}_N \), then 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 \( g = g, u = g^a, v = g^h, h = g, X_3 = X_3, e(g, h)^a = e(g^a X_2, h), \) and \( P_i \)'s decryption key share \( SK_i = (g^a X_2)^{f(i)} Z_{3i} \), and verification shadow \( V K_i = e(g, SK_i) \), for \( i = 1, \ldots, n \), where \( Z_{3i} \in \mathcal{G}_{p_2} \). Finally, \( D \) sends \( PK = (\mathcal{G}, \mathcal{G}_T, N, g, u, v, X_3, e(., h), e(g, h)^a), VK = (VK_1, \ldots, VK_n) \) to \( A \).

**Phase 1.** This phase is identical to Phase 1 of Game_5, with one difference in the DecapsulationShare query in that \( D \) chooses three random numbers \( \gamma, \zeta, \xi, \xi \in \mathbb{Z}_N \) and generates the partial decapsulation share of \( P_i \) with \( (c_1, c_2) \) as follows:

\[
\mu_{11} = SK_1 (u^t v)^{\gamma} (Z_2 X_3)^{\zeta},
\]
\[
\mu_{12} = g^{v} \cdot (Z_2 X_3^{\xi}),
\]
where \( \tau = H(c_1) \). Suppose that \( Z_2 = g^{\xi} X_3 = g_3^{\xi} \), \( (t, \sigma) \in \mathbb{Z}_N \), to verify the correctness, notice that we rewrite \( \mu_{11}, \mu_{12} \) as follows:

\[
\mu_{11} = SK_1 (u^t v)^{\gamma} (Z_2 X_3)^{\zeta} = SK_1 (u^t v)^{\gamma} g_3^{\zeta} \cdot g_3^{\xi} = SK_1 (u^t v)^{\gamma} \cdot g^{\xi} X_3 = \mu_{12} = g^{v} \cdot (Z_2 X_3^{\xi}) = g^{v} \cdot g^{\xi} X_3 = g^{v} \cdot V_3.,
\]
where \( V_3 = g_3^{\xi} \) and \( V_3 = g_3^{\xi} \). The form of \( \mu_{11}, \mu_{12} \) is the same as in Game_5’s.

**Challenge.** Once the adversary \( A \) ends phase 1, \( D \) can generate the challenge ciphertext as follows:

\[
c_1^* = g^b Y_2,
\]
\[
c_2^* = (g^b Y_2)^{a \bar{\tau}^* + b},
\]
\[
K_0 = T^c,
\]
where \( \tau = H(c_1) \). \( D \) now selects a random bit \( \delta \in \{0, 1\} \) and random session key \( K_1 \). \( D \) then sends \( (C^* = (c_1^*, c_2^*), K_3) \) to \( A \).

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

\[
c_1^* = g_2^\zeta Y_2 = g_2^\zeta g^\zeta,
\]
\[
c_2^* = (g_2^\zeta Y_2)^{a \bar{\tau}^* + b} = (g^b)^{a \bar{\tau}^* + b} \cdot (Y_2)^{a \bar{\tau}^* + b} = (g^b)^{a \bar{\tau}^* + b} \cdot (g^b)^{a \bar{\tau}^* + b} = (g^b)^{a \bar{\tau}^* + b} \cdot (g^b)^{a \bar{\tau}^* + b} = (g^b)^{a \bar{\tau}^* + b}.
\]

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

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

**Guess.** \( A \) outputs a guess bit \( \delta^* \in \{0, 1\} \) for \( \delta \).

If \( T = e(g, g)^{a \beta} \), \( D \) correctly simulates the game_5, since \( K_0 = T^c = e(g, g)^{a \beta} = e(g^a X_2, h)^{a \beta} = e(g, h)^{a \beta} \).

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

If \( A \) can distinguish Game_5 from Game_4 with a advantage \( \epsilon \), \( D \) uses the output of \( A \) to decide whether or not \( T = e(g, g)^{a \beta} \) with advantage \( \epsilon \).
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 DBDH 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>
<thead>
<tr>
<th>Scheme</th>
<th>Adaptive Corruption</th>
<th>Secure Model</th>
<th>Security Level</th>
<th>Security Reduction</th>
</tr>
</thead>
<tbody>
<tr>
<td>BMW'05</td>
<td>×</td>
<td>IND-CCA</td>
<td>SM</td>
<td>loose</td>
</tr>
<tr>
<td>IAHS'07</td>
<td>×</td>
<td>IND-CCA</td>
<td>ROM</td>
<td>tight</td>
</tr>
<tr>
<td>OURS</td>
<td>✓</td>
<td>IND-CCA</td>
<td>SM</td>
<td>tight</td>
</tr>
</tbody>
</table>

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

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.