Research Square

Preprints are preliminary reports that have not undergone peer review.
They should not be considered conclusive, used to inform clinical practice,
or referenced by the media as validated information.

Sender and Recipient Un-restricted, Non-interactive Group Key Exchange for Dynamic Group with Low-quality Network

Rui Zhang
East China Normal University
Lei Zhang
leizhang@sei.ecnu.edu.cn

East China Normal University
Yihuai Liang
Southwest Jiaotong University
Jiahuai Mao
Hong Kong Polytechnic University
Jianying Zhou
Singapore University of Technology and Design

Research Article

Keywords: non-interactive group key exchange, dynamic, fully collusion-resistance, standard model

Posted Date: June 22nd, 2026

DOI: https://doi.org/10.21203/rs.3.rs-9547370/v1

License: This work is licensed under a Creative Commons Attribution 4.0 International License.
Read Full License

Additional Declarations: No competing interests reported.

Sender and Recipient Un-restricted, Non-interactive Group Key Exchange for Dynamic Group with Low-quality Network

Rui Zhang[1,2], Lei Zhang[1*†], Yihuai Liang[3†], Jiahuai Mao[4†], Jianying Zhou[5†]

[1*]Department of Software Engineering, East China Normal University, Shanghai, 200062,China.
[2]Department of Computer Science and Artificial Intelligence, Southwestern University of Finance and Economics, Chengdu, 611130, China.
[3]Department of Information Science and Technology, Southwest Jiaotong University, Chengdu, 611756, China.
[4]Department of Computing, The Hong Kong Polytechnic University, Hong Kong,999077,China.
[5]Department of Computer Science,iTrust, Singapore University of Technology and Design, 487372, Singapore.

*Corresponding author(s). E-mail(s): leizhang@sei.ecnu.edu.cn;
Contributing authors: zhangrui0605@swufe.edu.cn;
liangyh@swjtu.edu.cn; jiahuai.mao@polyu.edu.hk;
jianying_zhou@sutd.edu.sg;
†These authors contributed equally to this work.

Abstract

Non-interactive group key exchange is a fundamental primitive for establishing secure communication channels in multi-party settings. However, existing protocols usually rely on computationally expensive tools, such as multi-linear maps and indistinguishability obfuscation, or lack efficient support for dynamic groups. To address these limitations, we propose a secure and efficient non-interactive group key exchange protocol built upon bilinear maps, which offers the following advantages over existing solutions. First, our protocol suits well for dynamic groups by enabling non-interactive key updates to accommodate

frequent membership changes. Second, it exhibits two desirable features for distributed applications: sender un-restriction, which permits any party to send a message, and recipient un-restriction, which allows a sender to select an arbitrary subset of members as recipients. Furthermore, our protocol generates constant-size ciphertexts, enhancing its efficiency and scalability. We formally prove that our protocol achieves fully collusion-resistance in the standard model under the k-Bilinear Diffie-Hellman Exponent (k-BDHE) assumption. Performance analysis confirms the protocol's practical efficiency.

Keywords: non-interactive group key exchange, dynamic, fully collusion-resistance, standard model

1 Introduction

Key exchange (KE) facilitates the establishment of a shared secret key, known as a session key, between two or more parties over an insecure channel [1, 2]. It has been widely utilized in various scenarios, such as online social networks [3] and video/radio calls[4]. In most KE protocols, parties engage in interactive communication to negotiate a session key through at least one round of interaction. However, interactive KE may not be suitable for certain applications. For example, in mobile ad-hoc networks (MANETs) [5–7], frequent interaction among parties is often impractical due to low-quality networks which are characterized by limited bandwidth, high latency, and/or unreliable connectivity. Additionally, the dynamic nature of the network topology in MANETs requires frequent updates to the session key. This can result in significant network delays and even protocol failures when using interactive KE protocols. Non-interactive key exchange (NIKE) offers a solution by enabling two or more parties, armed only with the public system parameters and each other's public key, to negotiate a session key without any interaction. By eliminating the communication complexity during session key establishment, NIKE holds great potential for applications in scenarios with low-quality networks.

Traditional NIKE protocols primarily suited for two-party settings (e.g., client and server communication) [1, 8], which inherently limits their real-world applicability. As most applications are group-oriented, designing NIKE protocols for multi-party environments has become an imperative necessity. However, existing NIKE protocols still face a few challenges when establishing a session key for group-oriented applications. Firstly, the adoption of current multi-party NIKE protocols [9–11] in real-world applications is impractical due to their reliance on heavy-weight cryptographic tools such as multilinear maps (MMPs), indistinguishability obfuscation (IO), and isogenise. Secondly, it is important to consider that in certain distributed systems, external entities have the desire to select preferred group members within a group for connection. For instance, in mobile ad-hoc networks, a node may wish to choose stable devices with good performance to obtain data. However, existing multi-party NIKE protocols do not support this communication mode, limiting their applicability in such scenarios.

1.1 Related WorK

The concept of two-party NIKE was first introduced in [1]. Later, instantiated two-party NIKE protocols were constructed [8, 12] based on different assumptions (e.g., factoring assumption and strong DH assumption). In the three-party setting, constructing NIKE was a major open problem until the Joux's work basing on the bilinear Diffie Hellman assumption over pairing groups [13], which could be regarded as the potential construction of three-party NIKE under a standard assumption. Although effective constructions of two/three-party NIKE protocols have been developed, there has not been an efficient construction of NIKE when the number of parties exceeds three.

Existing NIKE protocols for more than three parties rely on some cryptographic tools including MMPs[9, 14–16], IO [10, 17–19], and elliptic curve isogenies [11]. The earliest multi-party NIKE protocol, based on the hardness assumption of MMPs, was discussed in [14]. Since then, some MMPs-based multi-party NIKE protocols have been proposed [9, 15, 16]. Although multilinear maps enable the construction of multi-party NIKE protocols, the security of existing MMP candidates has been severely undermined by recent cryptanalysis [20]. The first IO-based multi-party NIKE protocol was introduced in [10]. Since then, several improved constructions have been proposed to enhance this paradigm. For instance, a multi-party NIKE scheme built upon ring key-homomorphic weak PRFs (RKHwPRFs) was presented in [21], while the one in [19] integrated Fully Homomorphic Encryption (FHE) to construct a protocol capable of supporting an arbitrary number of entities. Apart from the methods above, Boneh et al. described a mathematical framework based on isogenies to build multiparty NIKE [11]. However, instantiations of multi-party NIKE from isogenies incur heavy computational overhead, making it impractical in real-world scenarios.

Recently, researchers utilized fine-grained cryptography to construct multi-party NIKE protocols including four-party protocol in the generic group model [22] and multi-party protocol in the standard model [23]. However, these protocols require strong sub-exponential assumptions and depend on bounded models with strict adversarial resource constraints (both computational and storage), making them restricted for deployment in open network environments. In terms of round complexity, the most efficient KE for groups currently requires one interaction, relying on advanced primitives such as continuous group key agreement (CGKA) [24, 25] and contributory broadcast encryption (CBE) [26, 27]. Notably, CBE has drawn much focus and inspired many instantiations for various group applications [26, 28–31] , due to its sender unrestriction feature that allows any entity to be a sender without joining a group in advance. Specifically, by generating a common group encryption key and individual decryption keys, CBE-based KE allows any external sender to securely share a session key with an arbitrarily chosen subset of group members. However, these protocols still inherently require at least one round of interaction for key establishment.

1.2 Our Contribution

In this paper, we propose a non-interactive group key exchange protocol based on bilinear maps. By performing our protocol, multiple parties in group-oriented applications,

without any interaction among each other, are able to derive asymmetric group keys including a public group encryption key and each party's decryption key. Unlike existing multi-party NIKE protocols, our protocol negotiates different types of group keys instead of a single group session key. With the public group encryption key, any entity can establish a session with entities inside a group without needing to join the group in advance. This feature makes our protocol suitable for applications where entities may be geographically dispersed, such as across different time zones. Our multi-party NIKE protocol supports dynamic membership. When an entity joins or leaves a group, the asymmetric group keys can be updated accordingly but without introducing extra communication complexity. This feature implies significant importance for distributed group-oriented applications, especially those with dynamic topology of networks. In addition, our protocol considers the demand that an entity outside a group prefers to connect with those group members that has relatively stable performance. As for this, our protocol allows any entity to encrypt an session key under the public group encryption key for its selected group members within a group, and then every chosen group member could obtain the session key by decrypting the ciphertext with its own decryption key.

Another contribution of our work is to design a security model for our multi-party NIKE protocol and formalize its security definition. To the best of our knowledge, the CKS model in [32] was the classic security model for two-party NIKE, later extended to the m-CKS-heavy model in [8]. In fact, the m-CKS-heavy model strengthens the CKS model by capturing malicious behaviors, such as allowing an attacker to compromise an honest entity or reveal a certain session key. The security model for multi-party NIKE was first formally defined in [11], essentially extending the m-CKS-heavy model to multi-party settings, in which the indistinguishability of the session key was defined (this implies the confidentiality of the session key). In this paper, the security model is based on the model proposed in [11]. Slightly different, our model allows an attacker to designate any entity to join/leave a certain group corresponding to a session as our protocol supports dynamic membership. Our protocol ensures only the entity selected as the real recipient can obtain the session key even when an attacker obtains long-term private keys and decryption keys of entities who are not target recipients, which means the confidentiality of session key and fully resistant collusion are guaranteed simultaneously in our protocol.

2 Preliminaries

2.1 Complexity Assumption

The security of our protocol is reduced to the decision k-Bilnear Diffie-Hellman exponent (BDHE) assumption, which is first introduced in [33].

Decision k-BDHE problem: Given a bilnear map $\hat{e} : \mathbb{G} \times \mathbb{G} \rightarrow \mathbb{G}_T$. Let g be a generator of $\mathbb{G}$, $Q = g^h$ for unknown $h \in \mathbb{Z}_q$, and $\mathbb{X} = \{X_i = g^{\theta^i}\}_{\{i=1,2,\ldots,k,k+2,\ldots,2k\}}$ for unknown $\theta \in \mathbb{Z}_q^*$. An algorithm $\mathcal{D}$ that outputs $b \in \{0, 1\}$ has advantage ϵ in solving the decision k-BDHE problem if

$$|\Pr[\mathcal{D}(g, Q, X, Z_0) = 0] - \Pr[\mathcal{D}(g, Q, X, Z_1) = 0]| \geq \epsilon$$

where $Z_0 = \hat{e}(g^{\theta^{k+1}}, Q)$ and $Z_1 \in \mathbb{G}_T$ randomly. The decision k-BDHE assumption holds in $\mathbb{G}_T$ if no polynomial-time algorithm has advantage at least ϵ in solving the decision k-BDHE problem in $\mathbb{G}_T$.

2.2 Bilinear Maps

Our protocol is based on the bilinear maps [34]. Let $\mathbb{G}_1$ and $\mathbb{G}_T$ be two multiplicative groups of prime order q, and g be a generator of $\mathbb{G}_1$. A map $\hat{e} : \mathbb{G}_1 \times \mathbb{G}_1 \to \mathbb{G}_T$ is called a bilinear map if it satisfies the following conditions:

- Bilinearity: $\hat{e}(g^{\alpha}, g^{\beta}) = \hat{e}(g, g)^{\alpha\beta}$ for all $\alpha, \beta \in \mathbb{Z}_q^*$.
- Non-degeneracy: There exist $a \in \mathbb{G}_1, b \in \mathbb{G}_2$ such that $\hat{e}(a, b) \neq 1$.
- Computability: For any $a, b \in \mathbb{G}_1$, $\hat{e}(a, b)$ can be calculated efficiently.

3 Non-interactive Group Key Exchange Protocol

We first give a high-level description of our protocol for dynamic groups and then present the instantiated protocol.

3.1 High-level Description

This protocol is defined by the following algorithms:

- GlobeSetup: This algorithm is used to generate the global system parameter.
- KeyRegis: This algorithm allows each party in the open network environment to register with a trusted authority so as to get a long-term public-private key pair.
- KeyDerive: This algorithm allows multiple parties to form a group and negotiate a shared group encryption key and their respective decryption keys in a non-interactive way.
- KeyUpdate: This algorithm is used to update the group encryption key and each group member's decryption key once the group membership changes.
- Encrypt: This algorithm allows any sender knowing the group encryption key of a group to send an encrypted session key (also called the broadcast ciphertext) to some chosen members within the group (often called recipients).
- Decrypt: This algorithm allows a recipient to obtain the session key through decrypting the broadcast ciphertext using its decryption key.

3.2 The Concrete Protocol

Our concrete protocol comes as follows:

- GlobeSetup(1^{λ}): On input the security parameter 1^{λ}, it generates a public system parameter list params as follows: choose two cyclic multiplicative groups $\mathbb{G}_1, \mathbb{G}_T$ with prime order q, where $\mathbb{G}_1$ is generated by g; choose a bilinear map $\hat{e} : \mathbb{G}_1 \times \mathbb{G}_1 \to \mathbb{G}_T$, $u \in \mathbb{G}_1$ and generate $\mathbb{H} = \{h_1, h_2, ..., h_l\}$ by randomly selecting $h_i \in \mathbb{G}_1, 1 \leq i \leq l$; generate L_s tuples of the format $(\mathbb{A}_{\gamma}, \mathbb{B}_{\gamma}, \mathbb{K}_{\gamma})$, $1 \leq \gamma \leq L_s$. For each tuple, it corresponds to a group with the maximal group size n and is generated as follows:

– For $1 \leq i \leq n$, choose $\alpha_{i\gamma}, \beta_{i\gamma} \in \mathbb{Z}_q^*$, compute $A_{i\gamma} = g^{\alpha_{i\gamma}}$, $B_{i\gamma} = g^{\beta_{i\gamma}}$ and set $A_\gamma = \{A_{i\gamma}\}_{i\in\{1,...,n\}}$ and $B_\gamma = \{B_{i\gamma}\}_{i\in\{1,...,n\}}$.
– For $1 \leq i, j \leq n, i \neq j$, compute $K_{ij\gamma} = h_j^{\alpha_{i\gamma}} u^{\beta_{i\gamma}}$ and set $K_{j\gamma} = \{K_{ij\gamma}\}_{1\leq i\leq n, i\neq j}$, $K_\gamma = \{K_{j\gamma}\}_{1\leq j\leq n}$.

The global system parameters is denoted as $\mathsf{params} = (\mathbb{G}_1, \mathbb{G}_T, \mathbb{Z}_q^*, \hat{e}, g, q, u, \mathbb{H}, \{\mathbb{A}_\gamma, \mathbb{B}_\gamma, \mathbb{K}_\gamma\}_{1\leq\gamma\leq L_s})$. Assume there are totally N users in the open networks. Let $\mathbb{P} = \{P_1, P_2, ..., P_N\}$ denote all the users. Then we have a set $\mathbb{I} = \{1, 2, ..., N\}$ recording the index of each user in $\mathbb{P}$. This index is called user index.

- KeyRegis: Suppose that a party P with user index i' intends to obtain a long-term public-private key pair from TA.

  Case 1: KeyRegis$((\mathbb{A}_\gamma, \mathbb{B}_\gamma, \mathbb{K}_\gamma), i)$: For a user, the index within a group is called group index. In this case, P determines its group index as i and the group corresponds to a tuple $(\mathbb{A}_\gamma, \mathbb{B}_\gamma, \mathbb{K}_\gamma)$. Then, TA generates a long-term public-private key pair (PK_i, SK_i) for P: choose $a_i, b_i \in \mathbb{Z}_q^*$ randomly; compute $A_i = g^{a_i}, B_i = g^{b_i}, K_{ij} = h_j^{a_i} u^{b_i}$, $1 \leq j \leq n$; get $SK_i = h_i^{a_i} u^{b_i}$, $PK_i = (i, A_i, B_i, \{K_{ij}\}_{1\leq j\leq n, j\neq i})$.

  Case 2: KeyGen$(\mathbb{A}_\gamma, \mathbb{B}_\gamma, \mathbb{K}_\gamma)$: In this case, P doesn't know its group index, TA firstly generates n pairs of public-private keys $\{PK_l, SK_l\}$ as follows: for $1 \leq l \leq n$, select $a_l, b_l \in \mathbb{Z}_q^*$, get $A_l = g^{a_l}, B_l = g^{b_l}, K_{lj} = h_j^{a_l} u^{b_l}$, $1 \leq j \leq n$; for $1 \leq l \leq n$, set $SK_l = h_l^{a_l} u^{b_l}$, $PK_l = (l, A_l, B_l, \{K_{lj}\}_{1\leq j\leq n, j\neq l})$. Assume P is the first user to enroll with TA, TA assigns the first public-private key pair to P, which implies P occupies the first position in a group corresponding to $(\mathbb{A}_\gamma, \mathbb{B}_\gamma, \mathbb{K}_\gamma)$.
  For simplicity, we regulate KeyRegis in our protocol runs as in Case 1. We note in both cases, TA issues a certificate to each legitimate user so as to ensure the validity of the user's public key.
- KeyDerive$((\mathbb{A}_\gamma, \mathbb{B}_\gamma, \mathbb{K}_\gamma), i, SK_i, \mathbb{U}, \{PK_i\}_{i\in\mathbb{U}}, id_\pi)$:
  Assume there are t users intending to establish a group corresponding to $(\mathbb{A}_\gamma, \mathbb{B}_\gamma, \mathbb{K}_\gamma)$. Let π and id_π denote the index of the group and a unique identifier of this group. We note id_π can be randomly chosen by one of the users. Assume these users' indexes within the group form an index set $\mathbb{U} = \{1, ..., t\}$. For $i \in \mathbb{U}$, each user P with the index within the group i and its private-public key pair (SK_i, PK_i) performs the following steps to obtain a shared group encryption key Ω and a decryption key d_i:

  – For $i \in \mathbb{U}$, parse PK_i as $(i, A_i, B_i, \{K_{ij}\}_{1\leq j\leq n, j\neq i})$.
  – Compute $Y_1 = \prod_{i=1}^{t} A_i \prod_{i=t+1}^{n} A_{i\gamma}$, $Y_2 = \prod_{i=1}^{t} B_i \prod_{i=t+1}^{n} B_{i\gamma}$ and output the group encryption key $\Omega = (Y_1, Y_2)$.
  – For $1 \leq i \leq n$, set $\hat{dk}_i = \prod_{j=1}^{t, j\neq i} K_{ji} \prod_{j=t+1}^{n, j\neq i} K_{ji\theta}$.
  – Set $d_i = \hat{dk}_i SK_i$. If $\hat{e}(d_i, g) \stackrel{?}{=} \hat{e}(h_i, Y_1)\hat{e}(u, Y_2))$, output d_i as the decryption key; else, abort the algorithm.

  Apart from outputting (Ω, d_i), the algorithm also outputs (G_π, M_π^i). $G_\pi = (\pi, id_\pi, \Omega, st, \Delta)$ is used to describe some basic information about a group, which can be accessed publicly. We note st is an n-bit string initialized with all zero, which is used to record all the positions occupied by all the group members. That

is, if the i-th position is occupied by a group member, set $[st]_i = 1$. Δ denotes an index set recording all the group indexes of group members. $M_\pi^i = (\hat{dk}_1, ..., \hat{dk}_n, d_i)$ denotes the i-th group member's member information corresponding to the π-th group, which is stored in the member's local database.

- $\mathsf{KeyUpdate}(G_\pi, PK)$: The algorithm allows a user or group member with PK to join or leave a group with G_π in a non-interactive manner. The algorithm can be discussed in the following two cases:

  Join: Assume a user $P_{i'}$ wants to join a group with G_π as the I-th group member, and currently there are t group members that form an index set $\{1, ..., t\}$. Let PK_I denote the public key of $P_{i'}$. If $([st]_I = 1) \vee (t + 1 > n)$, the algorithm aborts; else, it performs as follow:

  - Parse PK_I as $(I, A_I, B_I, \{K_{Ij}\}_{1 \le j \le n, j \ne I})$;
  - Compute the group encryption key Ω and a decryption key d_I for the new I-th group member by invoking $\mathsf{KeyDerive}$ with inputs $((\mathbb{A}_\gamma, \mathbb{B}_\gamma, \mathbb{K}_\gamma)$, $I, SK_i, \mathbb{V}, \{PK_j\}_{j \in \mathbb{V}}, id_\pi)$, where $\mathbb{V} = \{1, ..., t + 1\}$ and $I \in \mathbb{V}$.
  - Update the old group encryption key Ω and the string st in G_π by setting $Y_1 = Y_1 A_I A_{I\gamma}^{-1}$, $Y_2 = Y_2 B_I B_{I\gamma}^{-1}$ and $[st]_I = 1$;
  - For $1 \le j \le t, j \ne I$, update each old member's member information M_π^j by setting $\hat{dk}_l = \hat{dk}_l K_{Il} K_{Il\gamma}^{-1}$, $1 \le l \ne I \le n$, and $d_j = d_j K_{Ij} K_{Ij\gamma}^{-1}$;
  - Add i' to Δ and output the new group information G_π.

  We note for the I-th new member, it accepts d_I as its decryption key iff the equation is satisfied: $\hat{e}(d_I, g) \stackrel{?}{=} \hat{e}(h_I, Y_1)\hat{e}(u, Y_2))$.

  Leave: Assume there are currently t group members who form an index set $\{1, ..., t\}$ in the π-th group, and the J-th group member with user index i' and public key PK_J wants to leave this group permanently. The algorithm performs as follows:

  - Parse PK_J as $(J, A_J, B_J, \{K_{Jj}\}_{1 \le j \le n, j \ne J})$;
  - Update the group encryption key Ω and the string st in G_π by setting $Y_1 = Y_1 A_J^{-1} A_{J\gamma}$, $Y_2 = Y_2 B_J^{-1} B_{J\gamma}$ and $[st]_J = 0$;
  - For $1 \le j \le (t - 1), j \ne J$, update the remaining member's member information M_π^j by setting $\hat{dk}_l = \hat{dk}_l K_{Jl\gamma} K_{Jl}^{-1}$, $1 \le l, J \le n, l \ne J$ and $d_j = d_j K_{Jj}^{-1} K_{Jj\gamma}$.
  - Remove i' from Δ and output the new group information G_π.

  For the rest of member, it accepts its new decryption key d_j iff $\hat{e}(d_j, g) \stackrel{?}{=} \hat{e}(h_j, Y_1)\hat{e}(u, Y_2))$.

- $\mathsf{Encrypt}(G_\pi, \Delta', \mathbb{U})$: Anyone knowing the public group information could run the algorithm. Assume a user P chooses a group with G_π and some group members within the group whose user index set is denoted as $\Delta' \subseteq \Delta$. Let $\mathbb{U}$ and $\mathbb{S} = \{i | \forall i \in \{1, ..., n\}, [st]_i = 1\}$ respectively denote the group index set of chosen group members and the group index set of all the group members in the π-th group, where $\mathbb{U} \subseteq \mathbb{S}$. On input $(G_\pi, \mathbb{U})$, the algorithm performs as follows:

  - Get a set $\bar{\mathbb{U}} = \mathbb{S} \setminus \mathbb{U}$ and compute $\hat{Y}_1 = Y_1 \prod_{i \in \bar{\mathbb{U}}} A_{i\gamma}$, $\hat{Y}_2 = Y_2 \prod_{i \in \bar{\mathbb{U}}} B_{i\gamma}$;
  - Choose ρ from $\mathbb{Z}_q^*$ at random and compute $C_1 = g^\rho$, $C_2 = \hat{Y}_1^{\ \rho}$;

– Get a set $\bar{\mathbb{S}} = \{i | \forall i \in \{1, ..., n\}, [st]_i = 0\}$ and compute $k = \hat{e}(u^{\rho}, \prod_{i \in \mathbb{S}} B_i \prod_{i \in \bar{\mathbb{U}}} B_{i\gamma} \prod_{i \in \bar{\mathbb{S}}} B_{i\gamma})$ as the session key;
– Output a pair (C_h, k), where $C_h = (C_1, C_2)$.

We note the Encrypt algorithm finally outputs a pair (C_h, k), where C_h is called the header and $k \in \mathbb{G}_T$ is essentially a message encryption key. In the sequel, once P shares a session key k with group members in $\mathbb{U}$, P will choose a message m from the message space and encrypt m under k by using a semantically-secure symmetric encryption scheme. In addition, for the convenience of decryption for chosen group members, a sender generally uploads the tuple $(P, id_{\pi}, \Delta', \mathbb{U}, C_h)$ to a platform(i.e., a server).

- Decrypt$(G_{\pi}, \Delta', i', \mathbb{U}, d_i, C_h)$: For any user $P_{i'}$ with user index i', if $i' \in \Delta'$, then $P_{i'}$ computes the same session key with the sender by invoking the algorithm. On input $(G_{\pi}, \Delta, \mathbb{U}, i', d_i, C_h)$, if $i \notin \mathbb{U}$, the algorithm outputs *null*; otherwise, it outputs the session key as follows:

  – Parse C_h as (C_1, C_2);
  – Compute $\hat{d}_i = d_i \prod_{l \in \hat{\mathbb{U}}} K_{li\gamma}$;
  – Compute $k = \hat{e}(\hat{d}_i, C_1)\hat{e}(h_i, C_2)^{-1}$.

  We then show our protocol satisfies correctness. That is, if a sender gets a pair (C_h, k) by invoking Encrypt with $(G_{\pi}, \Delta', \mathbb{U})$ as inputs. For each group member $(i \in \mathbb{U})$, it can compute the same session key k with the sender by invoking Decrypt algorithm with inputs $(G_{\pi}, \Delta', \mathbb{U}, i, d_i, C_h)$. We note the correctness of the protocol is guaranteed by the correctness of the equation in Decrypt which is shown below: We note the correctness of the protocol is guaranteed by the correctness of the equation in Decrypt which is shown below:

$$
\begin{aligned}
&\hat{e}(\hat{d}_i, C_1)\hat{e}(h_i, C_2)^{-1} \\
&= \hat{e}\Big(\prod_{l \in \mathbb{S}} K_l, g^{\rho}\Big) \cdot \hat{e}\Big(\prod_{l \in \bar{\mathbb{S}}} K_{li\gamma}, g^{\rho}\Big) \\
&\quad \cdot \hat{e}\Big(\prod_{l \in \bar{\mathbb{U}}} K_{li\gamma}, g^{\rho}\Big) \cdot \hat{e}\Big(h_i^{\rho}, Y_1 \prod_{l \in \bar{\mathbb{U}}} A_{l\gamma}\Big)^{-1} \\
&= \hat{e}\Big(\prod_{l \in \mathbb{S}} h_i^{a_l} u^{b_l}, g^{\rho}\Big) \cdot \hat{e}\Big(\prod_{l \in \bar{\mathbb{S}}} h_i^{\alpha_{l\gamma}} u^{\beta_{l\gamma}}, g^{\rho}\Big) \\
&\quad \cdot \hat{e}\Big(\prod_{l \in \bar{\mathbb{U}}} h_i^{\alpha_{l\gamma}} u^{\beta_{l\gamma}}, g^{\rho}\Big) \\
&\quad \cdot \hat{e}\Big(h_i^{\rho}, \prod_{l \in \mathbb{S}} A_l \prod_{l \in \bar{\mathbb{S}}} A_{l\gamma} \prod_{l \in \bar{\mathbb{U}}} A_{l\gamma}\Big)^{-1} \\
&= \hat{e}\Big(h_i^{\rho}, \prod_{l \in \mathbb{S}} A_l \prod_{l \in \bar{\mathbb{S}}} A_{l\gamma} \prod_{l \in \bar{\mathbb{U}}} A_{l\gamma}\Big)
\end{aligned}
$$

$$
\begin{aligned}
&\cdot \hat{e}\Big(u^{\rho}, \prod_{l\in\mathbb{S}} g^{b_l} \prod_{l\in\bar{\mathbb{S}}} g^{\beta_{l\gamma}} \prod_{l\in\bar{\mathbb{U}}} g^{\beta_{l\gamma}}\Big) \\
&\cdot \hat{e}\Big(h_i^{\rho}, \prod_{l\in\mathbb{S}} R_l \prod_{l\in\bar{\mathbb{S}}} A_{l\gamma} \prod_{l\in\bar{\mathbb{U}}} A_{l\gamma}\Big)^{-1} \\
&= \hat{e}\Big(u^{\rho}, \prod_{l\in\mathbb{S}} g^{b_l} \prod_{l\in\bar{\mathbb{S}}} g^{\beta_{l\gamma}} \prod_{l\in\bar{\mathbb{U}}} g^{\beta_{l\gamma}}\Big) \\
&= \hat{e}\Big(u^{\rho}, \prod_{l\in\mathbb{S}} B_l \prod_{l\in\bar{\mathbb{S}}} B_{l\gamma} \prod_{l\in\bar{\mathbb{U}}} B_{l\gamma}\Big)
\end{aligned}
$$

4 Security Analysis

We first design the security model for our protocol and then present the formal security proof.

4.1 Security Model

We have defined a user set $\mathbb{P} = \{P_1, P_2, ..., P_N\}$ and an user index set $\mathbb{I} = \{1, 2, ..., N\}$. It is known that any users who form a user set $\hat{\mathbb{U}}$ ($\hat{\mathbb{U}} \subseteq \mathbb{P}$) can establish a group in the non-interactive way. Each built group has an identity id_π and a session round ℓ, where π denotes it is the π-th group currently. For a group which is firstly formed by a group of users, the session round ℓ is set to be 1. If a user/member joins/leaves the group, the session round ℓ will increase by 1. In our security model, the group information describing each formed group is denoted as $G_\pi = (\pi, \ell, id_\pi, st, \Omega, \Delta)$, which corresponds to the ℓ-th session round of the π-th group. In the sequel, we give the security model of our G-NIKE. It is essentially a security game run between a challenger $\mathcal{C}$ and a probabilistic polynomial time (PPT) adversary $\mathcal{A}$. The security game consists of three phases as follows:
Initial: In this phase, $\mathcal{C}$ generates params by running $\mathsf{GlobeSetup}$ algorithm with a security parameter λ, and returns params to $\mathcal{A}$. $\mathcal{A}$ then submits a subset of $\mathbb{I}$ to $\mathcal{C}$, which is denoted as $\mathbb{U}$. This phase is used to simulate $\mathsf{GlobeSetup}$ algorithm.
Query: In the second phase, $\mathcal{C}$ answers the following types of queries from $\mathcal{A}$:

- Register$((\mathbb{A}_\gamma, \mathbb{B}_\gamma, \mathbb{K}_\gamma), i', i)$: This query is used to model $\mathsf{KeyRegis}$ algorithm, which prompts $\mathcal{C}$ to register a user selected by $\mathcal{A}$. In particular, $\mathcal{C}$ maintains an initially empty list $\mathbf{L}_u$. $\mathcal{A}$ chooses and supplies a user index i' from $\mathbb{I}$. Assume $\mathcal{A}$ submits the γ-th tuple corresponding the group and the group index i of this user. $\mathcal{C}$ first generates a public-private key pair (PK_i, SK_i), then adds (i', PK_i, SK_i) to $\mathbf{L}_u$. Finally, $\mathcal{C}$ replies PK_i to $\mathcal{A}$.
- Extract(i'): This query allows $\mathcal{A}$ to extract the long-term private key held by a user who has been registered. $\mathcal{A}$ first inputs an index i' of a user $P_{i'}$. If $i' \in \mathbb{U}$, $\mathcal{C}$ aborts; else, $\mathcal{C}$ recovers the item (i', PK_i, SK_i) from $\mathbf{L}_u$ and returns SK_i to $\mathcal{A}$.
- Execute$((\mathbb{A}_\gamma, \mathbb{B}_\gamma, \mathbb{K}_\gamma), i, SK_i, \hat{\mathbb{S}}, \{PK_j\}_{j\in\hat{\mathbb{S}}}, id_\pi, \ell)$:
  This query is used to model $\mathsf{KeyDerive}$ algorithm. $\mathcal{A}$ asks $\mathcal{C}$ to form a group for multiple users with a user index set $\mathbb{S} \subseteq \mathbb{I}$ and obtains the group encryption key and one of these users' decryption key. Assume the user $\mathcal{A}$ designates has the group

index i, the group index set for these users is $\hat{\mathbb{S}}$, and it is the ℓ-th session of the π-th group. If $\hat{\mathbb{S}} \wedge \mathbb{U} \neq \emptyset$, $\mathcal{C}$ aborts; else, $\mathcal{C}$ generates and replies the group encryption key Ω and the decryption key d_i held by the i-th group member in the group.

- Join(i', PK_i, G_π, I): This query is used to model Join sub-algorithm of KeyUpdate algorithm, which allows $\mathcal{A}$ to randomly choose a user who has been registered previously to join a group. Assume $\mathcal{A}$ selects a group with G_π that corresponds to an initial tuple $(\mathbb{A}_\gamma, \mathbb{B}_\gamma, \mathbb{K}_\gamma)$ and the I-th position in the π-th group to add a new group member with a user index i' and a corresponding public key PK_I. If the query is invoked successfully, $\mathcal{C}$ replies the updated G_π to $\mathcal{A}$.
- Leave(PK_I, G_π, I): This query is used to model Leave sub-algorithm of KeyUpdate algorithm, which allows $\mathcal{A}$ to choose any group member to leave a group permanently. Assume $\mathcal{A}$ selects a group with G_π and the I-th group member within the group, and the I-group member has a public key PK_I. If the query is invoked successfully, $\mathcal{C}$ replies the updated G_π to $\mathcal{A}$.
- Reveal(i', π, ℓ): This query allows $\mathcal{A}$ to obtain the decryption key held by any user who participates in the ℓ-th session of the π-th group. Assume the user has user index i' and group index i. If $i' \in \mathbb{U}$, $\mathcal{C}$ aborts; else, $\mathcal{C}$ returns the decryption key d_i held by i-th group member in the group.
- Test$(\mathbb{U}^*, id_{\pi^*}, \ell^*)$: In this query, $\mathcal{A}$ first chooses a target group with id_{π^*}, a target session round ℓ^* of the π^*-th group and a target user index set $\mathbb{U}^*$ ($\mathbb{U}^* \subseteq \mathbb{U}$) within the target group. Suppose the group information of the target group is denoted as $G_{\pi^*} = \{\pi^*, id_{\pi^*}, \ell^*, st, \Omega, \Delta\}$. On input $(\mathbb{U}^*, id_{\pi^*}, \ell^*)$, $\mathcal{C}$ tosses a coin $b \in \{0, 1\}$ firstly. If $b = 0$, $\mathcal{C}$ gets C_{h^*} and a real session key k_0 by invoking Encrypt algorithm; else, $\mathcal{C}$ selects a random session key k_1 from the session key space $\mathbb{G}_T$. At last, $\mathcal{C}$ replies (C_{h^*}, k_b) to $\mathcal{A}$.

Guess: In this phase, $\mathcal{A}$ submits $b' \in \{0, 1\}$ to $\mathcal{C}$. If $\mathcal{A}$ is able to distinguish a valid session key calculated by a set of users from a random element of the session key space, that is $b' = b$, then $\mathcal{A}$ wins the above game with advantage $Adv_{\mathcal{A}}$, where $Adv_{\mathcal{A}} = 2|Pr[b' = b] - 1|$.

Definition 1 Our dynamic group non-interactive key exchange protocol is semi-statically secure if for any PPT adversary $\mathcal{A}$ in the above game satisfying the following conditions, the advantage $Adv_{\mathcal{A}}$ of $\mathcal{A}$ to win the above game is negligible.

- $\mathcal{A}$ submits a user index set $\mathbb{U}$ after obtaining the public system parameter.
- Each query to Extract must be on an index i' outside $\mathbb{U}$.
- Each query to Reveal must have $\hat{\mathbb{U}} \wedge \mathbb{U} = \emptyset$.
- The query to Test must be on a subset $\mathbb{U}^*$ of $\mathbb{U}$.

We note that this semi-static security model based on the original CBE protocol's security model [26, 27]. However, this original security model is not suitable for non-interactive group key agreement. Besides, referring to the security model in [11], we define the security of our protocol, that is, indistinguishability of the session key.

4.2 Security Proof

In this section, we propose the following theorem and the corresponding proof to present that our protocol is semi-statically secure in above security game.

Theorem 1 *Assume that there are at most N groups which can be established by invoking our protocol, and for each group, there are at most L sessions that can be launched. If there exists an adversary $\mathcal{A}$ who wins the above security game with advantage ϵ, then there exists an algorithm to solve the decision k-BDHE problem with advantage $\frac{1}{NL}\epsilon$.*

[*Proof*] Suppose $\mathcal{C}$ is given an instance $(g, Q, Z, X_1, ..., X_k, X_{k+2}, ..., X_{2k})$ of the decision k-BDHE problem, where $X_i = g^{\theta^i}, i \in \{1, ..., k, k+2, ..., 2k\}$ with an unknown $\theta \in \mathbb{Z}_q$. We show how $\mathcal{C}$ can use $\mathcal{A}$ to determine whether Z equals to $\hat{e}(g^{\theta^{k+1}}, Q)$ or a uniform element in $\mathbb{G}_T$.
Initial: $\mathcal{C}$ generates params as follows: for $1 \leq j \leq n$, choose $\zeta_j \in \mathbb{Z}_q^*$ and set $h_j = g^{\zeta_j} X_j$; set $P = g^{\alpha^k} = X_k$; for $1 \leq \gamma \leq L_s$, generate $(\mathbb{A}_\gamma, \mathbb{B}_\gamma, \mathbb{K}_\gamma)$ that corresponds to the maximal group size n:

- If $i = 1$, select $\alpha_{1\gamma}, \beta_{1\gamma} \in \mathbb{Z}_q^*$ randomly and compute $A_{1\gamma} = g^{\alpha_{1\gamma}} \prod_{i=2}^{n} X_{k-i+1}^{-1}$, $B_{1\gamma} = g^{\beta_{1\gamma}} y_1$, set $K_{1j\gamma} = A_{1\gamma}^{\zeta_j} X_j^{\alpha_{1\gamma}} \prod_{i=2}^{n} X_{k-l+1+j}^{-1} P^{\beta_{1\gamma}}$ for $2 \leq j \leq n$, set $K_{11\gamma} = \perp$.
- Else ($2 \leq i \leq n$), select $\alpha_{i\gamma}, \beta_{i\gamma} \in \mathbb{Z}_q^*$ randomly, compute $A_{i\gamma} = g^{\alpha_{i\gamma}} X_{k-i+1}$, $B_{i\gamma} = g^{\beta_{i\gamma}}$, set $K_{ij\gamma} = A_{1\gamma}^{\zeta_j} X_j^{\alpha_{i\gamma}} X_{k-i+1+j} P^{\beta_{i\gamma}}$ for $2 \leq j \leq n$, set $K_{ii\gamma} = \perp$.

$\mathcal{C}$sets params$=(\mathbb{G}_1, \mathbb{G}_T, \mathbb{Z}_q^*, \hat{e}, g, q, P, \mathbb{H}, \{\mathbb{A}_\gamma, \mathbb{B}_\gamma, \mathbb{K}_\gamma\}_{1 \leq \gamma \leq L_s})$ and returns it to $\mathcal{A}$. After receving params, $\mathcal{A}$ submits a user index set $\mathbb{U} \subseteq \mathbb{I}$ to $\mathcal{C}$.

Assume there are at most N groups that have been formed by $\mathcal{C}$ and in each group, the maximal number of session rounds is L. We note $\mathcal{C}$ chooses a group from all N groups as a target group (assume the π-group)and a corresponding target session round ℓ in advance. We note if it's not the π-th group, then all the transcripts are consistent with that in the real protocol, which means $\mathcal{C}$ can answer all the following queries correctly. In other words, we only need to consider the queries associated with the target group.

Query: $\mathcal{C}$ answers the following queries from $\mathcal{A}$:
Register$((\mathbb{A}_\gamma, \mathbb{B}_\gamma, \mathbb{K}_\gamma), i', i)$: $\mathcal{C}$ maintains an initially empty list $\mathbf{L}_u$. To answer the query, $\mathcal{C}$ performs as follows:

- If there exists an item (i', PK_i, SK_i) on $\mathbf{L}_u$, return PK_i as the answer;
- Else, select a_i, b_i from $\mathbb{Z}_q^*$ and do the following:
  - If $i' \notin \mathbb{U}$, set $A_i = g^{a_i}$,$B_i = g^{b_i}$, $K_{ij} = h_j^{a_i} P^{b_i}$, for $1 \leq j \neq i \leq n$; set $PK_i = (i, A_i, B_i, \{K_{ij}\}_{1 \leq j \leq n, j \neq i})$, $SK_i = K_{ii}$; add the item (i', PK_i, SK_i) to $\mathbf{L}_u$ and return PK_i as the answer.
  - Else, if $i \neq n$, set $A_i = g^{a_i} X_{k-i+1}$, $B_i = g^{b_i}$, $K_{ij} = A_i^{\zeta_j} X_j^{a_i} X_{k-i+1+j} P^{b_i}$, $1 \leq j \neq i \leq n$, set $PK_i = (i, A_i, B_i, \{K_{ij}\}_{1 \leq j \leq n, j \neq i})$, $SK_i = \perp$; else, set $A_i = g^{a_i} \prod_{i=2}^{n} X_{k-i+1}^{-1}$, $B_i = g^{b_i} X_1$, $K_{ij} = A_i^{\zeta_j} X_j^{a_i} \prod_{i=2}^{n} x_{k-i+1+j}^{-1} P^{d_i}$, $1 \leq j \neq i \leq n$,

set $PK_i = (i, A_i, B_i, \{K_{ij}\}_{1\leq j\leq n, j\neq i})$, $SK_i = \perp$; add the item (i', PK_i, SK_i) to $\mathbf{L}_u$ and return PK_i as the answer.

Extract(i'): On receiving a user index i', $\mathcal{C}$ does the following: if $i' \in \mathbb{U}$, abort; else, recover the item (i', PK_i, SK_i) from $\mathbf{L}_u$ and return SK_i to $\mathcal{A}$.

Execute($(\mathbb{A}_\gamma, \mathbb{B}_\gamma, \mathbb{K}_\gamma), i, SK_i, \hat{\mathbb{S}}, \{PK_j\}_{j\in\hat{\mathbb{S}}}, id_\pi, \ell$): Assume currently it is the ℓ-th session of π-th group and the user index set is Δ that corresponds to a group index set $\hat{\mathbb{S}} = \{1, ..., t\}$. If $\hat{\mathbb{S}} \wedge \mathbb{U} \neq \emptyset$, $\mathcal{C}$ aborts; else, $\mathcal{C}$ does the following:

- Compute $Y_1 = \prod_{l=1}^{t} A_l \prod_{l=t+1}^{n} A_{l\gamma}$ and $Y_2 = \prod_{l=1}^{t} B_l$ $\prod_{l=t+1}^{n} B_{l\gamma}$ and get the group encryption key $\Omega = (Y_1, Y_2)$ of π-th group.
- Recover SK_i from $\mathbf{L}_u$, for $1 \leq i \leq n$, compute $\hat{dk}_i = \prod_{l=1}^{t, l\neq i} K_{li} \prod_{l=t+1}^{n, l\neq i} K_{li\gamma}$ and get the decryption key held by i-th group member $d_i = \hat{dk}_i SK_i$.
- Generate n-bit empty string st. For $1 \leq i \leq n$, if $i \in \hat{\mathbb{S}}$, set $st_i = 1$.
- For $1 \leq i \leq t$, get $M^i_{\pi,\ell} = \{i', i, \hat{dk}_1, ..., \hat{dk}_n, d_i\}$, corresponding the member information for each group.
- Return the group information of π-th group $G_\pi = (\pi, \ell, id_\pi, st, \Omega, \Delta)$.

After answering the query, $\mathcal{C}$ generates a list $\mathbf{T}_{\pi,\ell} = (G_\pi, \{M^i_{\pi,\ell}\}_{1\leq i\leq t})$, which corresponds to the ℓ-th session of the π-th group. We note by invoking the following Join or Leave query, the number of group members will increase or decrease correspondingly. If $\mathcal{C}$ sets $\mathbf{T}_{\pi,\ell} = \mathbf{T}_{\pi,\ell-1}$ in the sequel Join or Leave query, $\mathcal{C}$ does the following: 1) Replace G_π with the updated G_π; 2) For $i \in \{1, .., t\}$, set $M^i_{\pi,\ell} = M^i_{\pi,\ell-1}$.

Join(i', PK_i, G_π, I): Assume the current session round is ℓ, there exists t group members currently, the current group information is $G_\pi = \{\pi, id_\pi, \ell, st, \Omega, \Delta\}$, and the group index of all group members of the π-th group forms a group index set $\mathbb{V} = \{1, ..., t\}$. If $st[I] \neq \perp$ or $t + 1 > n$, $\mathcal{C}$ aborts; else, $\mathcal{C}$ first recovers the tuple (i', PK_i, SK_i) from $\mathbf{L}_u$, sets $\mathbf{T}_{\pi,\ell} = \mathbf{T}_{\pi,\ell-1}$, and then does the following:

- Parse PK_i as $(i, A_i, B_i, \{K_{ij}\}_{1\leq j\leq n, j\neq i})$.
- Update Ω by setting $Y_1 = Y_1 A^{-1}_{i\gamma} A_i$ and $Y_2 = Y_2 B^{-1}_{i\gamma} B_i$.
- For $j \in \mathbb{V}$, update $M^j_{\pi,\ell}$ by setting $\hat{dk}_l = \hat{dk}_l K^{-1}_{jl\gamma} K_{jl}$, for $1 \leq l \neq j \leq n$ and $d_j = d_j K^{-1}_{ji\gamma} K_{ji}$.
- Set $st[I] = 1$, and $M^I_{\pi,\ell} = \{I, \hat{dk}_1, ..., \hat{dk}_n, d_I\}$ and add $M^I_{\pi,\ell}$ to the list $\mathbf{T}_{\pi,\ell}$, where $d_I = \hat{dk}_i SK_i$.
- Add i' to Δ and return the updated G_π to $\mathcal{A}$.

Leave(PK_I, G_π, I): Assume the current session round is ℓ, there are t existing group members totally, and the group information is $G_\pi = \{\pi, id_\pi, \ell, st, \Omega, \Delta\}$. Assume the group index of all group members of the π-th group forms a group index set $\mathbb{V} = \{1, ..., t\}$. To update G_π, $\mathcal{C}$ first sets $\mathbf{T}_{\pi,\ell} = \mathbf{T}_{\pi,\ell-1}$ and performs as follows:

- Parse PK_I as $(I, A_I, B_I, \{K_{Ij}\}_{1\leq j\leq n, j\neq I})$.
- Update Ω by setting $Y_1 = Y_1 A_{I\gamma} A^{-1}_I$ and $Y_2 = Y_2 B_{I\gamma} B^{-1}_I$.
- For $i \in \mathbb{V}$, update $M^i_{\pi,\ell}$ by setting $\hat{dk}_l = \hat{dk}_l K_{Il\gamma} K^{-1}_{Il}$, for $1 \leq l \neq I \leq n$ and $d_i = d_i K_{Ii\gamma} K^{-1}_{Ii}$.
- Set $st[I] = \perp$ and remove $M^I_{\pi,\ell}$ from the list $\mathbf{T}_{\pi,\ell}$.

- Removes i' from Δ and return the updated G_π to $\mathcal{A}$.

$\mathsf{Reveal}(i', \pi, \ell)$: If $i' \notin \mathbb{U}$, $\mathcal{C}$ recovers $M^i_{\pi,\ell}$ from $\mathbf{T}_{\pi,\ell}$ and returns d_i to $\mathcal{A}$; Else, $\mathcal{C}$ this query.
$\mathsf{Test}(\mathbb{U}^*, id_{\pi^*}, \ell^*)$: $\mathcal{A}$ submits a target group with id_{π^*}, a target session round ℓ^* and a user index set $\mathbb{U}^* \subseteq \mathbb{U}$. Suppose $(\mathbb{A}_\gamma, \mathbb{B}_\gamma, \mathbb{K}_\gamma)$ is the initial tuple with the maximal group size n and corresponds to the target group. Then, the current group information is $G_{\pi^*} = \{\pi^*, id_{\pi^*}, \ell^*, \Omega^*, st^*, \Delta^*\}$. A group index set $\mathbb{S}^*$ can be got from st^* in G_{π^*}, where $\mathbb{S}^* = \{i | st^*[i] \neq \perp\}$. Based on $\mathbb{S}^*$, define $\bar{\mathbb{S}}^* = \{i | st^*[i] = \perp\}$ and $\bar{\mathbb{U}}^* = \mathbb{S}^* \setminus \mathbb{U}^*$. In this query, an abort event $\mathsf{Event\ 1}$ is defined. If the target group that $\mathcal{A}$ submits is not the π-th group or the target session round is not the ℓ-th session round, we say $\mathsf{Event\ 1}$ happens. If $\mathsf{Event\ 1}$ doesn't happen, $\mathcal{C}$ does the following:

- If $b = 0$, compute $k_0 = Z\hat{e}(g, Q)^{\sum_{i \in \mathbb{S}^*} a_i + \sum_{i \in (\mathbb{U}^* \cup \mathbb{S}^*)} \alpha_{i\gamma}}$; otherwise, choose a session key k_1 from $\mathbb{G}_T$ at random.
- Choose $b \in \{0, 1\}$ randomly and return k_b to $\mathcal{A}$.

Guess: $\mathcal{A}$ submits $b' \in \{0, 1\}$ to $\mathcal{C}$ as its answer.

We have known that $\mathcal{A}$'s advantage to win the above game is at least $Adv_{\mathcal{A}}$. To solve the decision k-BDHE problem, it requires $\mathcal{C}$ doesn't abort. That is, $\mathsf{Event\ 1}$ doesn't take place. It is easy to have $\Pr[\neg \mathsf{Event\ 1}] \geq \frac{1}{NL}$. Therefore, the advantage of $\mathcal{C}$ to solve the decision k-BDHE problem is at least $\frac{1}{NL} Adv_{\mathcal{A}}$.

□

5 Performance Evaluation

We first compare our protocol with lightweight and single-round multi-party key exchange protocols. Then we design simulations to show the efficiency of each algorithm in our protocol.

5.1 Comparison

In this section, we compare our protocol with existing lightweight and single-round multi-party key exchange protocols [26, 27, 30, 35, 36]. Among them, the one in [35] known as BGW protocol, is regarded as the collusion-resistant (one-pass) muilti-party key exchange with short private keys and ciphertexts. The protocol in [36] and the ones in [26, 27] are pratically efficient and provably secure key exchange protocols in multi-party settings, which are respectively based on AGKA and CBE.

In table 1, we compare the security and function properties of the protocols, and analyze whether our protocol and selected benchmark are provably secure in the standard model. As shown in Table 1, it is easy to see that only our protocol satisfies all the listed properties simultaneously. While the one in [30] is comparable with ours, one round interaction is still needed to generate the common group encryption key and each party's decryption key. The protocols in [26, 27, 35] are interactive and do not support dynamic membership. The work of Zhang et al. [36] supports most goals, such as key-escrow freeness, sender un-restriction, dynamicity, and fully collusion resistance.

Table 1 Comparison of Properties

Properties	Protocols				
	[35]	[36]	[26, 27]	[30]	Ours
Key-escrow Freeness	×	✓	✓	✓	✓
Sender un-restriction	✓	✓	✓	✓	✓
Recipient un-restriction	✓	×	✓	✓	✓
Dynamicity	×	✓	×	✓	✓
Fully Collusion Resistance	✓	✓	✓	✓	✓
Non-interactive	×	×	×	×	✓
Standard Model	✓	×	✓	✓	✓

Nevertheless, it falls short in providing recipient un-restriction and non-interactivity, and crucially, relies on the random oracle model for its security proof.

In table 2, we analyse the computational complexity of our protocol with those of others. It is noted that the computation cost of GlobeSteup algorithm is not analyzed. This is because the algorithm only needs to be performed once, affecting little on the whole efficency of these protocols. Besides, some operations that can be precomputed are not considered here. Let T_M /T_E denote the computational costs of a scalar multiplication/exponentiation in $\mathbb{G}_1$ and $\mathbb{G}_T$. Let T_e be the cost of a bilinear pairing and T_h be the cost of a hash-to-point operation [36]. Furthermore, let n be the maximum group size. For an encryption operation, let s be the number of members in the target group, and u be the number of recipients within that group ($u \leq s$). We define $s' = n - s$ as the number of non-members and $u' = s - u$ as the number of non-recipient members. Separately, t denotes the number of members in a group during a join or leave operation.

As shown in table 2, while the BGW protocol in [35] has the minimal computational costs compared with others, it only supports static settings. The ones in [26, 27] are static and exhibit quadratic complexity in both KeyRegis and KeyDerive algorithms. While the protocol in [36] achieves the lowest complexity for most algorithms, the computational cost of KeyUpdate algorithm grows linearly with the group size, hindering its scalability. It is easy to observe that our protocol features the most efficient KeyUpdate algorithm among all dynamic alternatives. Although in our protocol, KeyDerive algorithm has a quadratic complexity, it is executed only once, implying one-time setup cost. This design trade-off enables higher efficiency for subsequent and more frequent operations, such as KeyUpdate, making our protocol more practical for dynamic group environments. Further, our protocol demonstrates comparable efficiency in other essential algorithms such as KeyRegis, Encrypt and Decrypt.

5.2 Simulation

In this section, we simulate the running of the KeyRegis, KeyUpdate, Encrypt and Decrypt algorithm respectively. It is noted that the GlobeSetup algorithm affects a little on the whole efficiency of our protocol since it is invoked only once. The simulations

Table 2 Comparison of Computational Cost

Algorithms	Protocols				
	[35]	[36]	[26, 27]	[30]	Ours
KeyRegis	$O(n)(T_E)$	$O(1)T_E$	$O(n^2 + n)(T_E + T_M)$	$-^\star$	$O(n)(T_E + T_M)$
KeyDerive	$-^\dagger$	$O(n + t)(T_E + T_M) + O(1)(T_e + T_h)$	$O(n^2 + n)T_M$	$O(n)(T_E + T_M)$	$O(n^2 + n)T_M + O(1)T_e$
KeyUpdate (Join)	static	$O(n)(T_E + T_M) + O(1)T_h$	static	$O(n)(T_E + T_M)$	$O(1)(T_E + T_e) + O(n)T_M$
KeyUpdate (Leave)	static	$O(n)(T_M + T_E) + O(1)T_e$	static	$O(n)T_M$	$O(1)(T_E + T_e) + O(n)T_M$
Encrypt	$O(s)T_M + O(1)(T_E + T_e)$	$O(1)(T_E + T_h)$	$O(1)T_E + O(s')T_M$	$O(s + s' + u')T_M + O(1)(T_E + T_e)$	$O(s + s' + u')T_M + O(1)(T_e + T_E)$
Decrypt	$O(s)T_M + O(1)T_e$	$O(1)(T_e + T_E + T_h)$	$O(1)T_e + O(s')T_M$	$O(u')T_M + O(1)T_e$	$O(u')T_M + O(1)T_e$

‡: The aggregation of group encryption key and each group member's decryption key is not involved in [35].
⋆: The protocol in [30] does not involve the generation of public-private key pairs.

were run on a Ubuntu machine with an Intel Core i7-4790 at a frequency of 3.6 GHz by using cryptographic library MIRACL [37]. The security parameter was set to be 128 and a SSP curve with 128-bit security level was selected. The group size was set from 10 to 100, and the number of group members were set to be 80% of each group size. The recipients were chosen from existing group members randomly every time running the Encrypt algorithm. For simplicity, the operations that can be pre-computed were neglected in the simulations.

Fig. 1 presents the time cost of performing KeyRegis and KeyDerive. It is easy to see that the running time of both algorithms scales with the group size. However, the group size has a more significant impact on the running time of KeyDerive. When group size is 100, the time costs of KeyRegis and KeyDerive are respectively less than $200ms$ and $600ms$. Since KeyUpdate consists of Join and Leave sub-algorithms, then we measured the running time of both of them. As shown in Fig. 2, for an old group member (existing in the group), the execution time of Join increases linearly with group size. For a new group member wanting to join a group, the time cost of performing

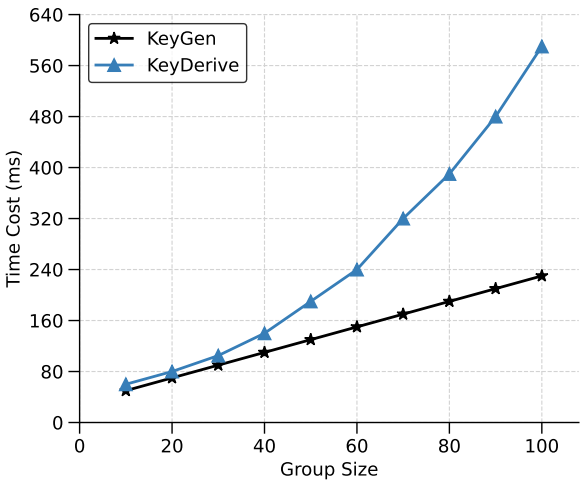


Fig. 1 Time cost of KeyRegis and KeyDerive

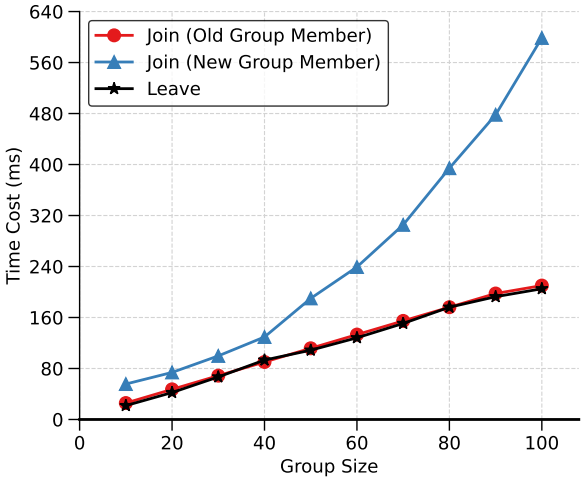


Fig. 2 Time cost of KeyUpdate

Join grows with group size exponentially. One can see that the time cost of running Leave approximately equals to that of running Join for an old group member. When the group size is 100, the overall execution time of Join/Leave is still acceptable (less than $200ms$ for an old member performing Join/Leave while less than $650ms$ for a new group member running Join). Hence, the KeyUpdate algorithm is efficient.

The time costs of running Encrypt and Decrypt are shown in Fig. 3. It is easy to see that the time cost of running Encrypt grows slowly with the group size. This is because the execution time of Encrypt is influenced by the number of recipients that increases correspondingly with the group size. Also, one can see that the time cost of running Decrypt remains constant for all group size. Overall, when the group size is

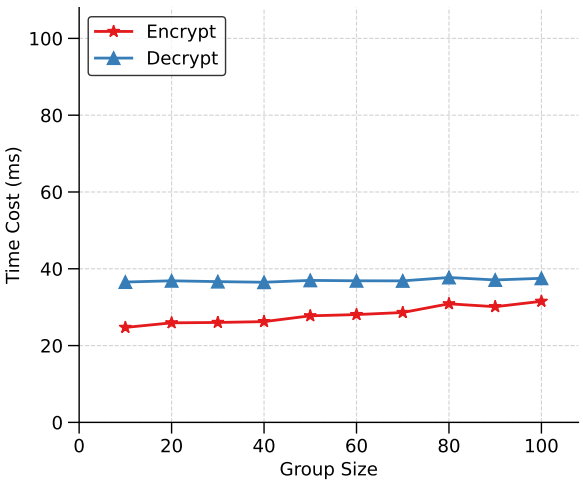


Fig. 3 Time cost of Encrypt and Decrypt

100, the time cost for performing Encrypt and Decrypt is less than $32ms$ and $38ms$ respectively. Therefore, both Encrypt and Decrypt are efficient.

6 Conclusion

In this paper, we propose a non-interactive key exchange protocol for secure communications in dynamic groups, which is based on bi-linear maps. This protocol allows multiple parties to derive two types of group keys (a public group encryption key and each party's decryption key) without any interaction among one another. Any party (even outside a group) establishes a secure channel with some/all group members of a group by encrypting a session key under the group encryption key, and the one selected by the party can decrypt the ciphertext and get the session key using its decryption key. This feature makes our protocol suitable for dynamic and distributed applications. Additionally, our protocol supports updating of group encryption key and each party's decryption key in an non-interactive way once any party outside a group or any group member joins or leaves the group. In the future, we plan to enhance our security model to be an adaptively secure one, and enable our non-interactive group key exchange protocol to be proven adaptively secure in the standard model.

Acknowledgements. This work was supported in part by the National Science Foundation of China under Grant 62372177.

References

[1] Diffie, W., Hellman, M.E.: New directions in cryptography. In: Democratizing Cryptography: the Work of Whitfield Diffie and Martin Hellman, pp. 365–390 (2022)

[2] Bresson, E., Chevassut, O., Pointcheval, D.: Provably secure authenticated group diffie-hellman key exchange. ACM Transactions on Information and System Security (TISSEC) **10**(3), 10 (2007)

[3] Cohn-Gordon, K., Cremers, C., Dowling, B., Garratt, L., Stebila, D.: A formal security analysis of the signal messaging protocol: K. cohn-gordon et al. Journal of Cryptology **33**(4), 1914–1983 (2020)

[4] Callas, J., Johnston, A., Zimmermann, P.: Zrtp: Media path key agreement for unicast secure rtp. RFC 6189 (2011)

[5] Hegland, A.M., Winjum, E., Mjolsnes, S.F., Rong, C., Kure, O., Spilling, P.: A survey of key management in ad hoc networks. IEEE Communications Surveys & Tutorials **8**(3), 48–66 (2006)

[6] Dalal, R., Singh, Y., Khari, M.: A review on key management schemes in manet. International Journal of Distributed and Parallel Systems **3**(4), 165 (2012)

[7] John, S.P., Samuel, P.: Self-organized key management with trusted certificate exchange in manet. Ain Shams Engineering Journal **6**(1), 161–170 (2015)

[8] Freire, E.S., Hofheinz, D., Kiltz, E., Paterson, K.G.: Non-interactive key exchange. In: Public-Key Cryptography–PKC 2013: 16th International Conference on Practice and Theory in Public-Key Cryptography, Nara, Japan, February 26–March 1, 2013. Proceedings 16, pp. 254–271 (2013). Springer

[9] Jia, H., Hu, Y., Wang, X.A., Liu, Z., Xiong, W.: Extensional schemes of multipartite non-interactive key exchange from multilinear maps. In: 2015 10th International Conference on P2P, Parallel, Grid, Cloud and Internet Computing (3PGCIC), pp. 771–774 (2015). IEEE

[10] Rao, V.: Adaptive multiparty non-interactive key exchange without setup in the standard model. Cryptology ePrint Archive (2014)

[11] Boneh, D., Glass, D., Krashen, D., Lauter, K., Sharif, S., Silverberg, A., Tibouchi, M., Zhandry, M.: Multiparty non-interactive key exchange and more from isogenies on elliptic curves. Journal of Mathematical Cryptology **14**(1), 5–14 (2020)

[12] Freire, E.S., Hesse, J., Hofheinz, D.: Universally composable non-interactive key exchange. In: Security and Cryptography for Networks: 9th International Conference, SCN 2014, Amalfi, Italy, September 3-5, 2014. Proceedings 9, pp. 1–20 (2014). Springer

[13] Joux, A.: A one round protocol for tripartite diffie–hellman. In: Algorithmic Number Theory: 4th International Symposium, ANTS-IV Leiden, The Netherlands, July 2-7, 2000. Proceedings 4, pp. 385–393 (2000). Springer

[14] Boneh, D., Silverberg, A.: Applications of multilinear forms to cryptography. Contemporary Mathematics **324**(1), 71–90 (2003)

[15] Pointcheval, D., Sanders, O.: Forward secure non-interactive key exchange. In: Security and Cryptography for Networks: 9th International Conference, SCN 2014, Amalfi, Italy, September 3-5, 2014. Proceedings 9, pp. 21–39 (2014). Springer

[16] Chen, Y., Huang, Q., Zhang, Z.: Sakai–ohgishi–kasahara identity-based non-interactive key exchange revisited and more. International Journal of Information Security **15**, 15–33 (2016)

[17] Khurana, D., Rao, V., Sahai, A.: Multi-party key exchange for unbounded parties from indistinguishability obfuscation. In: International Conference on the Theory and Application of Cryptology and Information Security, pp. 52–75 (2015). Springer

[18] Boneh, D., Zhandry, M.: Multiparty key exchange, efficient traitor tracing, and more from indistinguishability obfuscation. Algorithmica **79**(4), 1233–1285 (2017)

[19] Nassar, S., Waters, B.: Adaptive nike for unbounded parties. Cryptology ePrint Archive (2026)

[20] Cheon, J.H., Han, K., Lee, C., Ryu, H., Stehlé, D.: Cryptanalysis of the multilinear map over the integers. In: Annual International Conference on the Theory and Applications of Cryptographic Techniques, pp. 3–12 (2015). Springer

[21] Alamati, N., Montgomery, H., Patranabis, S.: Multiparty noninteractive key exchange from ring key-homomorphic weak prfs. In: Cryptographers' Track at the RSA Conference, pp. 333–359 (2023). Springer

[22] Afshar, A., Couteau, G., Mahmoody, M., Sadeghi, E.: Fine-grained non-interactive key-exchange: constructions and lower bounds. In: Annual International Conference on the Theory and Applications of Cryptographic Techniques, pp. 55–85 (2023). Springer

[23] Wang, Y., Su, C., Pan, J., Xu, C.: Fine-grained non-interactive key-exchange without idealized assumptions, and lower bounds. Cryptology ePrint Archive (2024)

[24] Auerbach, B., Cueto Noval, M., Pascual-Perez, G., Pietrzak, K.: On the cost of post-compromise security in concurrent continuous group-key agreement. In: Theory of Cryptography Conference, pp. 271–300 (2023). Springer

[25] Alwen, J., Auerbach, B., Noval, M.C., Klein, K., Pascual-Perez, G., Pietrzak, K., Walter, M.: Cocoa: Concurrent continuous group key agreement. In: Annual

International Conference on the Theory and Applications of Cryptographic Techniques, pp. 815–844 (2022). Springer

[26] Wu, Q., Qin, B., Zhang, L., Domingo-Ferrer, J., Farras, O., Manjon, J.A.: Contributory broadcast encryption with efficient encryption and short ciphertexts. IEEE Transactions on Computers **65**(2), 466–479 (2016)

[27] Wu, Q., Qin, B., Zhang, L., Domingo-Ferrer, J., Farras, O.: Bridging broadcast encryption and group key agreement. In: Advances in Cryptology–ASIACRYPT 2011: 17th International Conference on the Theory and Application of Cryptology and Information Security, Seoul, South Korea, December 4-8, 2011. Proceedings 17, pp. 143–160 (2011). Springer

[28] Zhang, L., Wu, Q., Domingo-Ferrer, J., Qin, B., Dong, Z.: Round-efficient and sender-unrestricted dynamic group key agreement protocol for secure group communications. IEEE Transactions on Information Forensics and Security **10**(11), 2352–2364 (2015)

[29] Li, J., Zhang, L.: Sender dynamic, non-repudiable, privacy-preserving and strong secure group communication protocol. Information Sciences **414**, 187–202 (2017)

[30] Zhang, L.: Key management scheme for secure channel establishment in fog computing. IEEE Transactions on Cloud Computing **9**(3), 1117–1128 (2019)

[31] Chen, T., Zhang, L., Choo, K.-K.R., Zhang, R., Meng, X.: Blockchain-based key management scheme in fog-enabled iot systems. IEEE Internet of Things Journal **8**(13), 10766–10778 (2021)

[32] Cash, D., Kiltz, E., Shoup, V.: The twin diffie–hellman problem and applications. Journal of cryptology **22**, 470–504 (2009)

[33] Boneh, D., Boyen, X., Goh, E.-J.: Hierarchical identity based encryption with constant size ciphertext. In: Annual International Conference on the Theory and Applications of Cryptographic Techniques, pp. 440–456 (2005). Springer

[34] Okamoto, T.: Cryptography based on bilinear maps. In: International Symposium on Applied Algebra, Algebraic Algorithms, and Error-Correcting Codes, pp. 35–50 (2006). Springer

[35] Boneh, D., Gentry, C., Waters, B.: Collusion resistant broadcast encryption with short ciphertexts and private keys. In: Annual International Cryptology Conference, pp. 258–275 (2005). Springer

[36] Zhang, R., Zhang, L., Choo, K.-K.R., Chen, T.: Dynamic authenticated asymmetric group key agreement with sender non-repudiation and privacy for group-oriented applications. IEEE Transactions on Dependable and Secure Computing **20**(1), 492–505 (2021)

[37] Scott, M.: Multiprecision integer and rational arithmetic c library. http://indigo.ie/~mscott (2024)