Research Article

A Certificateless Noninteractive Key Exchange Protocol with Provable Security

Xuefei Cao,1 Lanjun Dang,1 Yingzi Luan,2 and Wei You1

1School of Cyber Engineering, Xidian University, No. 2 South Taibai Rd., Xi’an 710071, China
2School of Telecommunication Engineering, Xidian University, No. 2 South Taibai Rd., Xi’an 710071, China

Correspondence should be addressed to Xuefei Cao; xfcao@xidian.edu.cn

Received 6 June 2020; Revised 16 July 2020; Accepted 14 August 2020; Published 28 August 2020

1. Introduction

Noninteractive key exchange (NIKE) protocols enable two users to establish a shared key without any interactions. In a NIKE, every user puts up his public key in a public directory, and two users can set up a shared key with the other’s public key and his own private key [1]. The earliest example of NIKE is Diffie–Hellman key exchange proposed in the seminal paper of [2]. In the protocol, Alice and Bob share a common group $G$ of order $q$ with generator $g$. Alice’s public key is of the form $g^x \in G$, and Bob’s public key is of the form $g^y \in G$. Then, Alice and Bob can establish a shared key $g^{xy}$ without any communications. NIKE is very useful to secure the communications where the communication delay matters, for example, the communications in the wireless networks where two terminals are far away from each other. Another example is the communication between the satellite and the earth. The distance between the satellite and the earth will increase the running time of a security protocol dramatically, and a NIKE protocol will reduce the protocol delay to the minimum because no interaction is needed between the earth and the satellite [3]. With NIKE protocols, two participants can establish a key and send encrypted message to its peer right away [4, 5].

Similar to NIKE protocols, public key encryption can also realize noninteractive communications. However, there are differences between the two. In a public key encryption system, anyone who wants to send the receiver Alice an encrypted message, only needs to know Alice’s public key and the system parameters. In a NIKE system, both parties should be enrolled in the system and have their private keys in order to establish a shared key. Furthermore, the encryption algorithm in a public key system is much slower than the encryption algorithm in a NIKE system because the latter could use the symmetric encryption algorithm once the shared key is set up.

In this paper, we propose a certificateless noninteractive key exchange protocol. No message exchange is required in the protocol, and this feature will facilitate the applications where the communication overhead matters, for example, the communications between the satellites and the earth. The public key certificate is removed as well as the key escrow problem using the certificateless public key cryptosystem. The security of the protocol rests on the bilinear Diffie–Hellman problem, and it could be proved in the random oracle model. Compared with previous protocols, the new protocol reduces the running time by at least 33.0%.
(4) The computation efficiency of the proposed protocol is improved compared with the available NIKE protocols.

The remaining part of this paper is organized as follows: Section 2 provides a research on the related work of NIKE protocols; Section 3 introduces the preliminaries, security definition, and security model; in Section 4, we introduce our scheme; Section 5 gives the security proof of our scheme. Section 6 gives the performance comparison; and Section 7 concludes the paper.

2. Related Works

According to the cryptosystems underlined, available NIKE protocols can be divided into three categories, i.e., certificate-based ones, identity-based ones, and certificateless ones. Certificate-based NIKE protocols employ the traditional certificate-based cryptography. Diffie—Hellman's noninteractive key exchange scheme falls into this category. In 2006, a certificate-based NIKE protocol was proposed using an elliptic curve [6], and a PKI-based security model was given. Later, Cash et al. proposed a stronger security model for NIKE, and a NIKE protocol using the twin Diffie—Hellman problem was also proposed [7]. In 2013, Freire et al. provided different security models for NIKE and studied the relationship between different security models. They also gave two NIKE constructions with provable security [8]. Hesse et al. proposed a NIKE with tight security reduction in [9].

In 1991, Maurer and Yacobi proposed an identity-based NIKE protocol based on a one-way trapdoor function [10]. However, it was pointed out that Maurer and Yacobi’s scheme was weak in security [11]. Later, Maurer and Yacobi improved their protocol [12], but the scheme was still proven insecure [13]. In 2000, Sakai et al. [14] proposed an ID-based NIKE protocol by introducing the bilinear pairings, but there were no formal security proofs. Dupont and Enge extended Sakai et al.’s scheme to a more general case and provided a security model for ID-based NIKE [15]. Paterson and Sirinivasan studied the relationship between ID-based NIKE and ID-based encryption and proposed an ID-NIKE scheme [16] and a security model stronger than Dupont and Enge’s model. An improved ID-NIKE with forward secrecy is proposed in [17].

To remove the inborn issue of key escrow with ID-based cryptosystem, Al-Riyami and Paterson proposed the certificateless cryptography [18]. In certificateless cryptography, a user's private key is generated both by the key generation center (KGC) and the user. Since, in the private key, there is a portion which is unknown to KGC, certificateless cryptography removes the key escrow problem. Compared with ID-based cryptosystem, certificateless cryptosystem maintains the former’s strength of lightweight public key management while achieves an improved level of security. Even the security authority is unable to know the secret established. In 2014, Sang et al. proposed a certificateless NIKE (CL-NIKE) protocol [19]. Later, Fu and Liu proposed another CL-NIKE protocol [20]. However, neither of the two protocols provided integrated security proof.

3. Preliminaries and Security Models

3.1. Preliminaries. Let \( G_1 \) be an additive group of order \( q \), where \( q \) is a large prime, and \( G_2 \) be a multiplicative group of the same order. Let \( P \) be an arbitrary generator of \( G_1 \); then, a bilinear pairing \( e \) is a map \( e : G_1 \times G_1 \rightarrow G_2 \) satisfying the following properties:

1. Bilinearity: given \( P \in G_1 \) and \( a, b \in \mathbb{Z}_q^* \), we have \( e(aP, bP) = e(P, P)^{ab} \).
2. Nondegeneracy: given \( P \) a generator of \( G_1 \), then \( Q = e(P, P) \) is a generator of \( G_2 \).
3. Computability: \( e(P_1, P_2) \) is efficiently computable for all \( P_1, P_2 \in G_1 \).

The following problem is assumed intractable in polynomial time [18]: bilinear Diffie—Hellman problem (BDHP): given \( (aP, bP, cP) \in G_1^3 \) for unknown \( a, b, c \in \mathbb{Z}_q^* \), compute \( e(P, P)^{abc} \).

3.2. Security Definition. In this section, we present the definition of CL-NIKE. A CL-NIKE scheme is defined by the following six algorithms:

1. Setup: this algorithm is run by the KGC once at the beginning to set up a certificateless key agreement system. It takes security parameter \( k \) and returns system parameters \( p \) and the master key of KGC. params are publicly authentic and the master-key is known only to the KGC.
2. Partial-private-key-extract: the algorithm generates the partial private key for system users. It is run by the KGC. It takes \( p \), master key, and entity’s identifier \( ID_A \) and outputs \( A \)'s partial private key \( D_A \). KGC sends \( D_A \) to \( A \) via a secret channel.
3. Set-secret-value: the algorithm is run by an entity \( A \). The algorithm returns \( A \)'s secret value \( x_A \).
4. Set-private-key: the algorithm is run by \( A \). It takes \( x_A \) and \( D_A \) and returns \( A \)'s private key \( S_A \).
5. Set-public-key: the algorithm is run by \( A \). It takes \( x_A \) and \( S_A \) and returns \( A \)'s public key \( P_A \).
6. Shared-key: on input params, \( A \)'s identifier \( ID_A \) and private key \( S_A \), and entity \( B \)'s identifier \( ID_B \) and public key \( P_B \), this algorithm outputs a shared key \( K_{AB} \) between \( A \) and \( B \).

3.3. Security Model. CK model [21] and eCK model [22] are the most widely used security models for authenticated key exchange (KE) protocols. However, they are not suitable for noninteractive key exchange protocols because CK model and eCK model provide security analysis to ephemeral secrets [23] but no ephemeral secrets are used in NIKE protocols. Therefore, a NIKE protocol needs its own security model.
Bernstein [6] and Cash et al. [7] first proposed the security models for NIKE protocols, respectively. Later, Dupont and Enge [15] and Paterson and Srinivasan [16] extended the security models in [6,7] from the certificate-based cryptosystem to the ID-based cryptosystem. Paterson’s security model is stronger than Dupont’s security model because Paterson’s model considers the security against the known session key attack by allowing the reveal query in the security model. Our security model follows Paterson’s model in [16]. Furthermore, we extend Paterson’s security model from ID-based cryptosystem to certificateless cryptosystem. Our security model considers the particular public/private key setting in the certificateless public key cryptosystem, and the case where an attacker replaces the legal user’s public key, and the case where the malicious KGC wants to break the shared key.

Now we define our security model of CL-NIKE protocols.

There are two types of adversaries to a CL-NIKE protocol, i.e., type I adversary \( \mathcal{A}_I \) and type II adversary \( \mathcal{A}_{II} \). Type I adversary simulates the ordinary attacker who is not able to get a user’s partial private key but is able to replace a legal user’s public key. \( \mathcal{A}_I \) is able to do this because there is no public key certificate in a CL-NIKE. Type II adversary simulates the malicious KGC who owns the system master key and hence knows every user’s partial private key but cannot replace the legal user’s public key because that is easily detected and could destroy KGC’s reputation. Consider two games, game I and game II between a challenger \( \mathcal{C} \) and \( \mathcal{A}_I \) and \( \mathcal{A}_{II} \), respectively. The security of a CL-NIKE protocol is defined via the two games:

**Game I:** the game is between \( \mathcal{A}_I \) and \( \mathcal{C} \).

Setup phase: in this phase, given security parameter \( k \), challenger \( \mathcal{C} \) obtains the system parameters params and master key. \( \mathcal{C} \) gives both params and master key to \( \mathcal{A}_I \).

Query phase: in this phase, the adversary \( \mathcal{A}_I \) can carry out the following queries in any order, and \( \mathcal{C} \) will answer the queries.

1. **Partial private key extraction:** \( \mathcal{A}_I \) chooses an entity with identifier \( ID_i \) and queries user \( I \)’s partial private key. \( \mathcal{C} \) runs the Partial-Private-Key-Extract to generate the partial private key \( D_i \) and returns it to \( \mathcal{A}_I \).
2. **Secret value extraction:** after receiving the query, \( \mathcal{C} \) runs the Set-Secret-Value algorithm and returns \( x_i \) to \( \mathcal{A}_I \).
3. **Private key extraction:** \( \mathcal{C} \) will call Partial-Private-Key-Extract and Set-Secret-Value in this query to obtain \( D_i \) and \( x_i \), and then, \( \mathcal{C} \) runs Set-Private-Key on \( D_i \) and \( x_i \) to generate the private key \( S_i \).
4. **Public key request:** \( \mathcal{C} \) first runs Set-Private-Key and keeps \( x_i \) for itself. Then, \( \mathcal{C} \) runs Set-Public-Key to generate \( ID_i \)’s public key \( P_i \) and returns it to \( \mathcal{A}_I \).
5. **Public key replacement:** \( \mathcal{A}_I \) will replace the public key of an entity with any value of its choice.
6. **Shared key revealing:** suppose the query is on \( ID_A \) and \( ID_B \), \( \mathcal{C} \) obtains \( S_A \) and \( P_B \), then runs Shared Key to obtain the \( K_{AB} \) between \( ID_A \) and \( ID_B \) and returns it to \( \mathcal{A}_I \).

Test phase: given a pair of identities \( ID_M \) and \( ID_N \), \( \mathcal{C} \) obtains \( K_{MN} \) as above. Then, \( \mathcal{C} \) selects at random \( b \in \{0,1\} \) and returns \( K_{MN} \) when \( b = 0 \) and \( \{0,1\} \) when \( b = 1 \). \( l \) is the length of the shared key.

Finally, \( \mathcal{A}_I \) outputs its guess \( b' \), and \( \mathcal{A}_{II} \) will win the game if \( b' = b \). \( \mathcal{A}_{II} \)’s advantage in the game is

\[
\text{Adv}_{\mathcal{A}_I}^{\text{Game I}} = \left| \Pr(b = b') - \frac{1}{2} \right|.
\]

We say the CL-NIKE scheme is secure against \( \mathcal{A}_I \) if \( \text{Adv}_{\mathcal{A}_I}^{\text{Game I}} \) is negligible.

**Game II:** the game is between \( \mathcal{A}_{II} \) and \( \mathcal{C} \).

Setup phase: in this phase, given security parameter \( k \), challenger \( \mathcal{C} \) obtains the system parameters params and master key. \( \mathcal{C} \) gives both params and master key to \( \mathcal{A}_{II} \).

Query phase: since \( \mathcal{A}_{II} \) has master key, it can compute the partial private key of any user. Then, in this phase, \( \mathcal{A}_{II} \) does not make the partial private key extraction query. Also, since \( \mathcal{A}_{II} \) is not allowed to replace the user’s public key, it does not make the public key replacement query. In game II, \( \mathcal{A}_{II} \) makes the queries and \( \mathcal{C} \) answers as follows:

1. **Secret value extraction:** on receiving the query on \( ID_1 \), \( \mathcal{C} \) runs the Set-Secret-Value and returns \( x_1 \) to \( \mathcal{A}_{II} \).
2. **Private key extraction:** if user \( I \)’s partial private key has not been computed, \( \mathcal{C} \) first computes \( I \)’s partial private key using master key, then it calls the Set-Secret-Value to obtain \( x_1 \). At last, \( \mathcal{C} \) runs the Set-Private-Key to obtain user \( I \)’s private key and returns it to \( \mathcal{A}_{II} \).
3. **Public key request:** on receiving this query, \( \mathcal{C} \) first runs Set-Secret-Value to obtain \( x_1 \) and then it runs Set-Public-Key to obtain the public key and returns it to \( \mathcal{A}_{II} \).
4. **Shared key revealing:** suppose this query is made on \( ID_A \) and \( ID_B \), \( \mathcal{C} \) runs corresponding algorithms to obtain the \( S_A \) and \( P_B \), and then, it computes \( K_{AB} \) by running the SharedKey and returns \( K_{AB} \) to \( \mathcal{A}_{II} \).

Test phase: given a pair of identities \( ID_M \) and \( ID_N \), \( \mathcal{C} \) obtains \( K_{MN} \) as above. Then, \( \mathcal{C} \) selects at random \( b \in \{0,1\} \) and returns \( K_{MN} \) when \( b = 0 \) and \( \{0,1\} \) when \( b = 1 \). \( l \) is the length of the shared key.

Finally, \( \mathcal{A}_{II} \) outputs its guess \( b' \), and \( \mathcal{A}_{II} \) will win the game if \( b' = b \). \( \mathcal{A}_{II} \)’s advantage in the game is

\[
\text{Adv}_{\mathcal{A}_{II}}^{\text{Game II}} = \left| \Pr(b = b') - \frac{1}{2} \right|.
\]

We say the CL-NIKE scheme is secure against \( \mathcal{A}_{II} \) if \( \text{Adv}_{\mathcal{A}_{II}}^{\text{Game II}} \) is negligible.
A CL-NIKE scheme is secure if it is secure both against \( \mathcal{A}_1 \) and \( \mathcal{A}_2 \).

### 4. Protocol Description

In this section, we propose a new CL-NIKE protocol. The new protocol includes six algorithms, as described in Section 3.2, and each algorithm is as follows:

(i) Setup: given system security parameter \( k \), KGC does the following:

1. Outputs \( G_1 \) and \( G_2 \) and \( e \) satisfying the definitions in Section 3.1.
2. Chooses an arbitrary generator \( P \) of \( G_1 \).
3. Selects the master key \( s \in \mathbb{Z}_q^* \) at random and computes the system public key \( P_0 = sP \).
4. Selects two hash functions \( H_i: \{0,1\}^* \rightarrow G_1 \) and \( H_s: \{0,1\}^* \rightarrow \{0,1\}^l \).
5. Publishes \([G_1,G_2,q,e,P,P_0,H_1,H_2]\) as system parameters and keeps \( s \) secretly.

(ii) Partial-Private-Key-Extract: given a user \( A \)'s identifier \( ID_A \), KGC computes \( Q_A = H_1(ID_A) \) and \( D_A = sQ_A \). \( D_A \) is \( A \)'s partial private key. KGC returns \( D_A \) to \( A \) via a secure channel.

(iii) Set-Secret-Value: \( A \) selects at random \( x_A \in \mathbb{Z}_q^* \) and sets \( x_A \) as his secret value.

(iv) Set-Private-Key: \( A \)'s private key is \( S_A = (x_A,D_A) = (x_A,sH_1(ID_A)) \).

(v) Set-Public-Key: \( A \) computes \( X_A = x_AP \) and \( Y_A = x_AP_0 \) and his public key is \( P_A = (X_A,Y_A) \).

(vi) SharedKey: to set up a shared key with \( B \), \( A \) first verifies whether \( B \)'s public key is of the correct form, i.e., \( A \) verifies if the equation \( e(X_B,P_0) = e(Y_B,P) \) holds. If the verification is correct, \( A \) computes the shared secrets:

\[
K^1_A = e(D_A + x_A Y_B, Q_B), \\
K^2_A = x_A Y_B.
\]

The shared key is

\[
K_{AB} = H_2(ID_A, ID_B, K^1_A, K^2_A).
\]

To compute the shared key, \( B \) first verifies \( A \)'s public key by checking if the equation \( e(X_A,P_0) = e(Y_A,P) \) holds. If the verification is correct, \( B \) computes the shared secrets:

\[
K^1_B = e(Q_A + x_B X_A, D_B), \\
K^2_B = x_B X_A.
\]

The shared key is

\[
K_{BA} = H_2(ID_A, ID_B, K^1_B, K^2_B).
\]

The correctness of the protocol could be guaranteed because

\[
K^1_A = e(D_A + x_A Y_B, Q_B) = e(Q_A + x_A x_B P, sQ_B) = e(Q_A + x_B X_A, D_B) = K^1_B, \\
K^2_A = x_A Y_B = x_A x_B P_0 = x_B Y_A = K^2_B, \\
K_{AB} = H_2(ID_A, ID_B, K^1_A, K^2_A) = H_2(ID_A, ID_B, K^1_B, K^2_B) = K_{BA}.
\]

The procedure of the SharedKey algorithm is illustrated in Figure 1.

### 5. Security Proof

In this section, we prove the security of our protocol in the random oracle model based on the security model in Section 3. We have the following theorem:

**Theorem 1.** If BDH problem is hard, our CL-NIKE protocol is secure in the random oracle model.

The theorem can be proven via Lemma 1 and Lemma 2.

**Lemma 1.** If there exists a type I adversary \( \mathcal{A}_1 \) which wins game I against our CL-NIKE protocol, then BDH problem can be solved with nonnegligible probability in the random oracle model.

**Proof.** Suppose a challenger \( \mathcal{C} \) is given an instance \((aP,bP,cP) \in G_1^3\) and is tasked to compute \( e(P,P)^{ab} \). If there is a type I adversary \( \mathcal{A}_1 \) which wins the game I against the proposed protocol, then \( \mathcal{C} \) can solve the BDH problem by controlling the queries with \( \mathcal{A}_1 \).

In the setup phase, \( \mathcal{C} \) chooses the system parameters \([G_1,G_2,q,e,P,H_1,H_2]\), and \( \mathcal{C} \) sets the system public key \( P_0 = cP \) with the master-key \( c \) unknown to \( \mathcal{C} \).

In the query phase, hash functions \( H_1 \) and \( H_2 \) are modeled as random oracles. Without generality, we suppose that there are \( n_1 \) users in the system, and \( \mathcal{A}_1 \) chooses \( ID_A \) and \( ID_B \) in the test phase. We also suppose \( \mathcal{A}_1 \) is allowed to ask \( n_1 \) queries. During this phase, \( \mathcal{C} \) answers every query as follows:

1. \( H_1 \) query: \( \mathcal{A}_1 \)'s \( H_1 \) query is of the form \((ID_A, h_1, h_1 P)\). \( \mathcal{C} \) maintains an \( H_1 \)-list with each entry of the form \([ID_1, h_1, h_1 P]\). On receiving \( \mathcal{A}_1 \)'s query, \( \mathcal{C} \) first searches \( H_1 \)-list whether there is an entry indexed by \( ID_A \). If there is, \( \mathcal{C} \) returns \( h_1 P \) to \( \mathcal{A}_1 \). If there is not, the following holds:

   a. If \( ID_A = ID_A \), \( \mathcal{C} \) inserts \([ID_A, \bot, aP]\) in the list and returns \( aP \) to \( \mathcal{A}_1 \).
   b. If \( ID_A = ID_B \), \( \mathcal{C} \) inserts \([ID_B, \bot, bP]\) in the list and returns \( bP \) to \( \mathcal{A}_1 \).
   c. If \( ID_A \notin [ID_1, ID_B] \), \( \mathcal{C} \) chooses at random \( h_1 \in \mathbb{Z}_q^* \), returns \( h_1 P \) to \( \mathcal{A}_1 \), and inserts \([ID_A, h_1, h_1 P]\) to the \( H_1 \)-list.

2. \( H_2 \) query: \( \mathcal{A}_1 \)'s \( H_2 \) query is of the form \([ID_j, ID_A, K^1_{ij}, K^2_{ij}]\). \( \mathcal{C} \) maintains an \( H_2 \)-list with each entry the form \([ID_j, ID_A, K^1_{ij}, K^2_{ij}, h^1_{ij}]\). On
receiving $\mathcal{A}_1$’s query, $\mathcal{C}$ first searches $H_2$-list whether there is an entry indexed by $(ID_i, ID_j)$. If there is an item indexed by $(ID_i, ID_j)$, $\mathcal{C}$ returns the corresponding $h_{ij}^2$ to $\mathcal{A}_1$. Or else $\mathcal{C}$ chooses at random $h_{ij}^2 \in [0,1]^l$, returns $h_{ij}^2$ to $\mathcal{A}_1$, and inserts $\{ID_i, ID_j, K_{ij}^1, K_{ij}^2, h_{ij}^2\}$ to $H_2$-list.

(3) Partial private key extraction: $\mathcal{C}$ maintains a $\mathcal{H}$-list of user’s keys. $\mathcal{H}$-list is of the form $\{ID_i, x_i, D_i, X_i, Y_i, \text{tag}\}$. $(x_i, D_i)$ are $I$’s private key, and $(X_i, Y_i)$ are $I$’s public key. If tag = 0, it means that $I$’s public key has not been replaced. If tag = 1, it means that $I$’s public key has been replaced, and the corresponding $(X_i, Y_i)$ is the replaced ones.

On receiving $\mathcal{A}_1$’s query on $ID_i$, $\mathcal{C}$ checks $\mathcal{H}$-list; if there is an entry indexed by $ID_i$ and $D_i \neq -$, $\mathcal{C}$ returns the corresponding $D_i$. If no entry indexed by $ID_i$ or $D_i = -$, $\mathcal{C}$ does the following:

(a) If $ID_i \notin \{ID_A, ID_B\}$, $\mathcal{C}$ checks $H_1$-list with $ID_i$ for $h_{ij}^1$. If there is no entry indexed by $ID_i$ in $H_1$-list, $\mathcal{C}$ first makes $H_1$ query. Then $\mathcal{C}$ computes $D_i = h_{ij}^1cP$. If no entry is indexed by $ID_i$ in $\mathcal{H}$-list, $\mathcal{C}$ inserts $\{ID_i, D_i, -, -,-\}$ to it; if there is an entry indexed by $ID_i$ but $D_i = -$, $\mathcal{C}$ updates $\mathcal{H}$-list, $\mathcal{C}$ only inserts $D_i$. Finally, $\mathcal{C}$ returns $D_i$ to $\mathcal{A}_1$.
(b) If $ID_i \in \{ID_A, ID_B\}$, $\mathcal{C}$ aborts the game.

(4) Secret value extraction: on receiving such a query on $ID_i$, $\mathcal{C}$ checks the $\mathcal{H}$-list. If there is an item indexed by $ID_i$ and $x_i \neq -$, $\mathcal{C}$ returns $x_i$ to $\mathcal{A}_1$. If there is not an item indexed by $ID_i$ or $x_i = -$, $\mathcal{C}$ runs the Set-Secret-Value algorithm to obtain $x_i$. If there is not an entry indexed by $ID_i$ in $H_1$-list, $\mathcal{C}$ adds $\{ID_i, x_i, -,-,-\}$ to it. If there is an entry indexed by $ID_i$ but $x_i = -$, $\mathcal{C}$ only adds $x_i$ to the entry. Finally, $\mathcal{C}$ returns $x_i$ to $\mathcal{A}_1$.

(5) Private key extraction: the query is on $ID_i$. If $ID_i \in \{ID_A, ID_B\}$, $\mathcal{C}$ aborts the game. Otherwise, $\mathcal{C}$ checks the $\mathcal{H}$-list. If there is an item indexed by $ID_i$, $x_i \neq -$, and $D_i \neq -$, $\mathcal{C}$ returns $(x_i, D_i)$ to $\mathcal{A}_1$. If there is not such an item or if there is an item indexed by $ID_i$, but at least one value of $x_i$ and $D_i$ equals $-$, according to the value needed, $\mathcal{C}$ runs Set-Secret-Value to obtain $x_i$ and queries the partial private key extraction oracle to obtain $D_i$. If there is no item indexed by $ID_i$, $\mathcal{C}$ adds $\{ID_i, x_i, D_i, -,-,-\}$ to the $\mathcal{H}$-list. If there is an item indexed by $ID_i$, but $(x_i, D_i)$ in the item is incomplete, $\mathcal{C}$ adds the missed value(s) to the item. Finally, $\mathcal{C}$ returns $(x_i, D_i)$ to $\mathcal{A}_1$.

(6) Public key request: on receiving such a query on $ID_i$, $\mathcal{C}$ checks the $\mathcal{H}$-list. If there is an item indexed by $ID_i$ and $(X_i, Y_i) \neq -$, $\mathcal{C}$ returns $(X_i, Y_i)$. If there is not an item indexed by $ID_i$, $\mathcal{C}$ runs the Set-Secret-Value to obtain $x_i$, computes $(X_i, Y_i)$, and then adds $\{ID_i, x_i, -, X_i, Y_i, 0\}$ to the $\mathcal{H}$-list. If there is an item indexed by $ID_i$, but $(X_i, Y_i) = -$, $\mathcal{C}$ first checks if $x_i = -$. If $x_i = -$, $\mathcal{C}$ runs the Set-Secret-Value algorithm to obtain $x_i$, computes $X_i, Y_i$, and adds $x_i, X_i, Y_i$, tag = 0 to the item. If $x_i \neq -$, $\mathcal{C}$ computes $(X_i, Y_i)$, inserts $X_i, Y_i$, tag = 0 in the item. Finally, $\mathcal{C}$ returns $(X_i, Y_i)$ to $\mathcal{A}_1$.

(7) Public key replacement: without generality, we assume that the query is of the form $\{ID_A, X'_i, Y'_i\}$, where $e(X'_i, cP) = e(Y'_i, P)$. On receiving such a query, $\mathcal{C}$ sets the entry indexed by $ID_i$ as $\{ID_i, D_i, X'_i, Y'_i, 1\}$ in the $\mathcal{H}$-list.

(8) Shared key revealing: $\mathcal{A}_1$’s query is on $(ID_j, ID_j)$. $\mathcal{C}$ aborts the game if $(ID_j, ID_j) = (ID_A, ID_B)$. Otherwise, $\mathcal{C}$ queries $ID_j$’s private key and $ID_j$’s public key, computes the shared key according to the protocol description, and returns $K_{AB}$ to $\mathcal{A}_1$.

In the test phase, $\mathcal{A}_1$’s query is on $(ID_j, ID_j)$. If $(ID_j, ID_j) \neq (ID_A, ID_B)$, $\mathcal{C}$ aborts the game. Else, $\mathcal{C}$ chooses uniformly $[0,1]^l$ and returns it to $\mathcal{A}_1$.

If $\mathcal{A}_1$ makes the correct guess, then he must have queried $H_2$ oracle. $\mathcal{C}$ searches the $H_2$ list for the entry $\{ID_A, ID_B, K_{AB}, K_{AB}, h_{AB}\}$. Suppose $B$’s public key has been replaced with $(X_B, Y_B')$, then we have that

$$K_{AB}^1 = e(D_A + x_A Y_B', Q_B) = e(acP + x_A Y_B', bP), \quad (9)$$

so $e(P, P)^{abc} = K_{AB}^1 \cdot e(K_{AB}^2, -bP).$

Suppose $\mathcal{A}_1$’s probability of success is $\text{Adv}_{\mathcal{A}_1}^{\text{Game1}}$, then $\mathcal{C}$’s probability of success is at least $(\text{Adv}_{\mathcal{A}_1}^{\text{Game1}})^2/((h_1^2 h_2^2))$. □

**Lemma 2.** Suppose $H_1$ and $H_2$ are random oracles. If a type II adversary $\mathcal{A}_1$ can win the game II against the proposed CL-
NIKE protocol, then the BDH problem can be solved with nonnegligible probability.

Proof. Suppose there is a challenger $C$ who is given an instance of $(aP, bP, cP) \in G_1^3$ and is tasked to compute $e(P, P)^{abc}$. If there is a type II adversary $\mathcal{A}_I$, then $C$ can solve the BDH problem by controlling the queries with $\mathcal{A}_I$.

In the setup phase, $C$ chooses the system parameters $\{G_1, G_2, q, e, P, H_1, H_2\}$ and the master key $s$ and computes the system public key $P_0 = sp$. $C$ gives the system parameters $\{G_1, G_2, q, e, P, H_1, H_2\}$ as well as the master key $s$ to $\mathcal{A}_I$.

In the query phase, hash functions $H_1$ and $H_2$ are modeled as random oracles. Without generality, we suppose that there are $n_1$ users in the system, and $\mathcal{A}_I$ chooses $ID_A$ and $ID_B$ to ask the test query. We also suppose $\mathcal{A}_I$ is allowed to ask $n_2$ $H_2$ queries. The ways that $C$ answers $\mathcal{A}_I$’s queries are as follows:

(1) $H_1$ query: $\mathcal{A}_I$’s $H_1$ query is of the form $(ID_i)$. $C$ maintains an $H_1$-list with each entry of the form $(ID_i, h_i^1, h_i^2P)$. On receiving $\mathcal{A}_I$’s query, $C$ first checks $H_1$-list. If there is an item indexed by $ID_i$, $C$ returns the corresponding $h_i^1P$ to $\mathcal{A}_I$. If there is not such an item, the following holds:

(a) If $ID_i = ID_B$, $C$ inserts $(ID_B, \bot, bP)$ in the list and returns $bP$.

(b) If $ID_i \neq ID_B$, $C$ chooses uniformly $h_i^1 \in \mathbb{Z}_q^*$, returns $h_i^1P$, and inserts $(\{ID_i, h_i^1, h_i^2\})$ to the $H_1$-list.

(2) $H_2$ query: $\mathcal{A}_I$’s $H_2$ query is of the form $(ID_i, ID_j, K_{ij}^1, K_{ij}^2)$. $C$ maintains an $H_2$-list with each entry the form $(ID_i, ID_j, K_{ij}^1, K_{ij}^2)$. On receiving $\mathcal{A}_I$’s $H_2$ query, $C$ first checks $H_2$-list. If there is an item indexed with $(ID_i, ID_j, K_{ij}^1, K_{ij}^2)$, then $C$ returns the corresponding $h_i^2P$. Else, $C$ chooses $h_i^2 = \epsilon h_i^2(0, 1)^t$, returns $h_i^2$ to $\mathcal{A}_I$, and inserts $(ID_i, ID_j, K_{ij}^1, K_{ij}^2, h_i^2)$ to $H_2$-list.

(3) Secret value extraction: $C$ maintains a $\mathcal{X}$-list of the form $(ID_i, x_i, D_i)$. On receiving $\mathcal{A}_I$’s query on $ID_i$, $C$ first checks whether there is an entry indexed by $ID_i$ in the $\mathcal{X}$-list. If there is, $C$ returns $x_i$. Otherwise, $C$ runs the Set-Secret-Value algorithm to obtain $x_i$, adds $(ID_i, x_i, \bot, \bot)$ to the $\mathcal{X}$-list, and returns $x_i$ to $ID_i$. We note here that it is impossible that there is an item indexed by $ID_i$ but $x_i = \bot$ because $D_i$ is known by $\mathcal{A}_I$ and the entry must be generated by asking $x_i$.

(4) Private key extraction: $\mathcal{A}_I$’s private key extraction query is on $ID_i$. If $ID_i \in \{ID_A, ID_B\}$, $C$ aborts the game. Else $C$ searches the $\mathcal{X}$-list, if there is an item indexed by $ID_i$ and $x_i \neq \bot$, $D_i \neq \bot$, and $C$ returns $(x_i, D_i)$. If there is not an item indexed by $ID_i$, $C$ runs the Set-Secret-Value to get $x_i$ and computes $D_i$ using $s$, then $C$ inserts $(ID_i, x_i, D_i, \bot, \bot)$ to the $\mathcal{X}$-list. If there is an item indexed by $ID_i$, but $D_i$ is missing, $C$ computes $D_i$ using $s$ and adds $D_i$ to the $\mathcal{X}$-list. Finally, $C$ returns $(x_i, D_i)$ to $\mathcal{A}_I$.

(5) Public key request: on receiving such a query on $ID_i$, $C$ checks the $\mathcal{X}$-list. If there is an entry listed by $ID_i$, and $(X_2, Y_2) \neq \bot$, $C$ returns $(X_2, Y_2)$. Otherwise, $C$ does the following:

(a) If $ID_i \notin \{ID_A, ID_B\}$ and there is not an item indexed by $ID_i$, $C$ runs the Set-Secret-Value to obtain $x_i$ and computes $X_i = x_iP$ and $Y_i = x_iSP$, then $C$ adds $(ID_i, x_i, X_i, Y_i)$ to the $\mathcal{X}$-list. If $ID_i \notin \{ID_A, ID_B\}$ and there is an entry indexed by $ID_i$, then the entry must be generated when $\mathcal{A}_I$ queries $ID_i$’s private key. $C$ obtains $x_i$ from the entry, computes $X_i = x_iP$ and $Y_i = x_iSP$, and adds $(X_i, Y_i)$ to the entry.

(b) If $ID_i = ID_A$, $C$ sets $X_A = aP$ and computes $Y_A = saP$. Then, $C$ adds $(ID_A, X_A, Y_A)$ to the $\mathcal{X}$-list.

(c) If $ID_i = ID_B$, $C$ sets $X_B = bP$ and computes $Y_A = scP$. Then, $C$ adds $(ID_B, X_B, Y_B)$ to the $\mathcal{X}$-list.

Finally, $C$ returns $(X_i, Y_i)$ to $\mathcal{A}_I$.

(6) Shared key revealing: $\mathcal{A}_I$’s query is on $(ID_i, \bot)$. If $(ID_i, \bot) = (ID_A, \bot)$, then $C$ aborts the game. Else, $C$ queries $ID_i$’s private key and $ID_i$’s public key, computes the shared key according to the protocol description, and returns $K_i$ to $\mathcal{A}_I$.

In the test phase, $\mathcal{A}_I$ chooses $(ID_i, ID_j)$ as query. If $(ID_i, ID_j) \neq (ID_A, ID_B)$, $C$ aborts the game. Else, $C$ chooses uniformly $[0, 1]^t$ and returns it to $\mathcal{A}_I$.

If $\mathcal{A}_I$ makes the correct guess, then he must have queried $H_2$ oracle. $C$ searches the $H_2$ list for the entry $(ID_A, ID_B, K_{AB}^1, K_{AB}^2, h_{AB})$. We have that $K_{AB} = e(D_A \cdot x_AY_B, Q_{AB}) = e(sh_AP + sacbP, bP) = e(sh_AP, bP)e(sacbP, bP)$, and then, $C$ can output $e(P, P)^{abc} = (K_{AB}^1)^{1/2} \cdot e(h_i^1P, bP)$. $\mathcal{A}_I$’s probability of success is $\text{Adv}_{\mathcal{A}_I}^{\text{Game II}}$, then $C$’s probability of success is at least $\left((\text{Adv}_{\mathcal{A}_I}^{\text{Game II}})/n_i^2n_j^2\right)$.

We also note here that, in a CL-NIKE protocol, a participant $ID_A$ has the private key or long-term secret of the form $(x_A, DA)$, where $x_A$ is $ID_A$’s secret number, and $DA$ is $ID_A$’s private secret key. A CL-NIKE protocol allows the leakage of $x_A$ or $DA$. In the above security proof, the security against a leaked $x_A$ is modeled by game I, which allows an attacker to obtain $x_A$ via the public key replacement query. The security against a leaked $DA$ is modeled by game II, which gives the attacker the master key or the private key of every participant in the system.

A CL-NIKE protocol does not allow the leakage of the $x_A$ or $DA$ or the private key. This is very different from authenticated KE protocols because authenticated KE protocols use ephemeral keys [24, 25], and the security of the shared key could be guaranteed by the ephemeral keys even
if both participants’ private keys are lost. In order to realize the noninteractive key establishment, there is no ephemeral keys used in NIKE protocols; thus, the leakage of private keys is not allowed in NIKE protocols. However, compared with certificate-based NIKE protocols [7] or identity-based NIKE protocols [15, 16], our CL-NIKE protocol allows the leakage of part of the private key, and this results in the improved security.

6. Performance Analysis

In this section, we compare our protocol with Sang et al.’s protocol [19] and Fu and Liu’s protocol [20] in terms of computation complexity and running time.

The computation complexity is compared in terms of complex operations including pairing, scalar multiplication, and hash to \( G_1 \) point. We neglect operations including ordinary hash functions, point addition, and integer computations because the running time of these operations is trivial compared with those complex operations. We compare how many complex operations are employed in every protocol. In the comparison, \( P \) stands for a pairing operation, \( S \) stands for a scalar multiplication, and \( H \) stands for a map to \( G_1 \) point hash. To evaluate the running time of a protocol, we first implement the complex cryptographic operations and then add up to get the overall running time.

We implement the complex operations with Miracle Version 7.0 [26]. We use the Tate pairing defined over supersingular elliptic curve \((E/F_p)\); \( y^2 = x^3 + x \) with embedding degree 2. \( q \) is a 160-bit Solinas prime \( q = 2^{159} + 2^{17} + 1 \) and \( p \) a 512-bit prime satisfying \( p + 1 = 12q \). The evaluation is carried out in a PC with Intel Core i7-6700 CPU at 2.8 GHZ and 8.0 GB memory. The operation system is Windows 10. The operation time of every complex operation is listed in Table 1.

The comparison results are given in Table 2. From the comparison, we can see that our protocol reduces the running time by limiting the pairing operation times. Compared with Sang et al.’s protocol, the running time is reduced by 34.75%; compared with Fu-Liu’s protocol, the running time is reduced by 33.00%.

7. Conclusion

In this paper, we propose a certificateless noninteractive key exchange protocol. The protocol requires no interaction; thus, the communication delay is minimized. Moreover, the computation efficiency is improved because the pairing operation times are reduced. Compared with existing CL-NIKE protocols, our protocol improves the computation efficiency by at least 33.00%. The security of the protocol is based on the bilinear Diffie–Hellman problem and could be proved in the random oracle model.

Data Availability

The data used to support the findings of this study are included within the article.

Conflicts of Interest

The authors declare that they have no conflicts of interest.

Acknowledgments

The authors would like to thank the National Defense Pre-Research Plan for the 13-th Five Years Project (no. 90407180012), National Science Foundation of China (no. 61771361), and Scientific Plan Project of Shaanxi Province (no. 2020JQ-319) for funding.

References

[1] E. S. V. Freire, J. Hesse, and D. Hofheinz, “Universally composable non-interactive key exchange,” Lecture Notes in Computer Science, Springer, Berlin, Germany, pp. 1–20, 2014.
[2] W. Diffie and M. Hellman, “New directions in cryptography,” IEEE Transactions on Information Theory, vol. 22, no. 6, pp. 644–654, 1976.
[3] R. Du, Z. Zhou, and H. Milan, “Non-interactive key exchange protocol for space dtm,” Computer Engineering, vol. 42, no. 4, pp. 137–142, 2016.
[4] M. Qi and J. Chen, “An enhanced authentication with key agreement scheme for satellite communication systems,” International Journal of Satellite Communications and Networking, vol. 36, no. 3, pp. 296–304, 2018.
[5] Y. Wei, F. Wei, and C. Ma, “Certificateless non-interactive key exchange protocol without pairings,” in Proceedings of the International Conference on Security and Cryptography, pp. 31–42, Pune, India, October 2014.
[6] D. J. Bernstein, “Curve25519: new diffie-hellman speed records,” Public Key Cryptography-PKC 2006, Springer, Berlin, Germany, pp. 207–228, 2006.
[7] D. Cash, E. Kiltz, and V. Shoup, “The twin diffie-hellman problem and applications,” Journal of Cryptology, vol. 22, no. 4, pp. 470–504, 2009.
[8] E. S. V. Freire, D. Hofheinz, E. Kiltz, and K. G. Paterson, “Non-interactive key exchange,” Public-Key Cryptography-PKC 2013, Springer, Berlin, Germany, pp. 254–271, 2013.
[9] J. Hesse, D. Hofheinz, and L. Kohl, “On tightly secure non-interactive key exchange,” Lecture Notes in Computer Science, Springer, Berlin, Germany, pp. 65–94, 2018.
[10] U. Maurer and Y. Yacobi, “Non-interactive public-key cryptography,” in Proceedings of the Eurocrypt, pp. 498–507, Brighton, UK, April 1991.

### Table 2: Efficiency comparison of different protocols.

| Protocol          | Computation complexity | Running time (in milliseconds) |
|-------------------|------------------------|-------------------------------|
| Sang et al.’s [19] | \( 5P + 2S + 1H \)    | 112.99                        |
| Fu and Liu’s [20]  | \( 5P + S + 2H \)     | 110.04                        |
| Our protocol      | \( 3P + 2S + H \)     | 73.73                         |
[11] C. H. Lim and P. J. Lee, "Modified Maurer-Yacobi’s scheme and its applications," *Advances in Cryptology-AUSCRYPT’92*, Springer, Berlin, Germany, pp. 308–323, 1993.

[12] U. M. Maurer and Y. Yacobi, "A non-interactive public-key distribution system," *Designs, Codes and Cryptography*, vol. 9, no. 3, pp. 305–316, 1996.

[13] U. Maurer and K. Dennis, “A note on the weakness of the maurer-yacobi squaring method,” Tech. Rep. TI 15/99, TU Darmstadt, Darmstadt, Germany, 1999.

[14] R. Sakai, K. Ohgishi, and M. Kasahara, "Cryptosystems based on pairing," *The 2000 Symposium on Cryptography and Information Security*, pp. 26–28, Springer, Berlin, Germany, 2000.

[15] R. Dupont and A. Enge, "Provably secure non-interactive key distribution based on pairings," *Discrete Applied Mathematics*, vol. 154, no. 2, pp. 270–276, 2006.

[16] K. G. Paterson and S. Srinivasan, "On the relations between non-interactive key distribution, identity-based encryption and trapdoor discrete log groups," *Designs, Codes and Cryptography*, vol. 52, no. 2, pp. 219–241, 2009.

[17] R. Steinwandt and A. Suárez Corona, "Identity-based non-interactive key distribution with forward security," *Designs, Codes and Cryptography*, vol. 64, no. 1-2, pp. 195–208, 2012.

[18] S. S. Al-Riyami and K. G. Paterson, "Certificateless public key cryptography," *Advances in Cryptology-ASIACRYPT 2003*, Springer, Berlin, Germany, pp. 452–473, 2003.

[19] Y. Sang, L. Zhang, L. You, and Z. Li, "Two non-interactive key agreement protocols under certificateless scenarios," *International Journal of Advancements in Computing Technology*, vol. 4, no. 6, pp. 331–337, 2012.

[20] X. Fu and X. Liu, "A non-interactive key agreement protocol under certificateless scenarios," *Journal of Cryptologic Research*, vol. 1, no. 4, pp. 334–340, 2014.

[21] R. Canetti and H. Crawczyk, "Analysis of key exchange protocols and their usage in building secure channels," in *Advances in Cryptology-Eurocrypt 2001*, LNCS 2045, pp. 453–474, Springer-Verlag, Berlin, Germany, 2001.

[22] B. LaMacchia, K. Lauter, and A. Mitya, "Stronger security of authenticated key exchange," in *Proceedings of Provsec 2007*, Springer-Verlag, Wollongong, Australia, pp. 1–16, October 2007.

[23] D. Abbasinezhad-Mood and M. Nikooghadam, "An anonymous ecc-based self-certified key distribution scheme for the smart grid," *IEEE Transactions on Industrial Electronics*, vol. 65, no. 10, pp. 7996–8004, 2018.

[24] D. Abbasinezhad-Mood, A. Ostad-Sharif, M. Nikooghadam, and S. M. Mazinani, "A secure and efficient key establishment scheme for communications of smart meters and service providers in smart grid," *IEEE Transactions on Industrial Informatics*, vol. 16, no. 3, pp. 1495–1502, 2020.

[25] D. Abbasinezhad-Mood, A. Ostad-Sharif, S. M. Mazinani, and M. Nikooghadam, "Provably-secure escrow-less chebyshev chaotic map-based key agreement protocol for vehicle to grid connections with privacy protection," *IEEE Transactions on Industrial Informatics*, 2020.

[26] Miracl Library, 2015, https://www.miracl.com/.