Certificateless and Identity based Authenticated Key Exchange Protocols

Saikrishna Badrinarayanan(✉) and C. Pandu Rangan

Department of Computer Science and Engineering,
Indian Institute of Technology Madras, Chennai, India
bsaikrishna7393@gmail.com, prangan@cse.iitm.ac.in

Abstract. Designing efficient key agreement protocols is a fundamental cryptographic problem. In this paper, we first define a security model for key agreement in certificateless cryptography that is an extension of earlier models. We note that the existing pairing free protocols are not secure in our model. We design an efficient pairing-free, single round protocol that is secure in our model based on the hardness assumption of the Computational Diffie Hellman (CDH) problem. We also observe that previously existing pairing-free protocols were secure based on much stronger assumptions such as the hardness of the Gap Diffie Hellman problem. We use a restriction of our scheme to design an efficient pairing-free single round identity based key agreement protocol that is secure in the id-CK+ model based on the hardness assumption of the CDH problem. Additionally, both our schemes satisfy several other security properties such as forward secrecy, resistance to reflection attacks etc.

Keywords: Certificateless cryptography · Identity-based cryptography · Key exchange · Random oracle

1 Introduction

Symmetric key cryptography is a paradigm in which both encryption and decryption is done using the same key unlike asymmetric system in which each user maintains a public key and a private key. Symmetric key cryptography is in general more efficient than an asymmetric system. However, the main disadvantage of symmetric key cryptography is the establishment of the shared secret key between the entities that want to communicate. A secure way of setting up the shared secret key is mandatory. In this work, we focus on key exchange protocols in the identity based and certificateless paradigm. Several key exchange protocols have been designed in these paradigms [4,13,16,17].

2 Previous Work and Our Contribution

2.1 Certificateless Cryptography

Several protocols and security models have been proposed for certificateless authenticated key exchange (CLAKE). The strongest security model is the one

M.S. Obaidat and A. Holzinger (Eds.): ICETE 2014, CCIS 554, pp. 255–277, 2015.
DOI: 10.1007/978-3-319-25915-4_14

proposed by Lippold et al. [12], which is based on the Canetti Krawczyk model for key agreement. In this paper, we propose a security model that is an extension of the one proposed by Lippold et al. Our model considers an active adversary-one who can tamper with any message that is being exchanged within the network. In a real world scenario, active adversaries are very much present making this an important consideration towards the security of protocols. Since pairing is an extremely costly mathematical operation, it hampers the efficiency of the system and so, we focus only on schemes which are pairing-free. Several pairing-free protocols [6,8,15,18] were proposed but most of them are either based on a weaker security model or have subsequently been broken. Two pairing-free protocols proposed by Yang et al. [18] and Sun et al. [15] are based on the Lippold et al. model. However, we observe that both these protocols are not secure in our definition. In particular, an active adversary can modify the ephemeral components and prevent the users from being able to compute the same shared secret without them realising that they are indeed not computing the same secret. The main advantage of our proposed scheme is that we prove the security of our protocol based on the hardness assumption of the Computational Diffie Hellman problem. We observe that all previously existing pairing-free key agreement protocols are proven secure based on much stronger assumptions like the Gap Diffie Hellman assumption. Another important property is the number of rounds in the protocol. Lesser the number of rounds, greater the efficiency of the protocol. Our proposed scheme is single round and hence can be implemented asynchronously while multiple round protocols need to be implemented synchronously and require both the parties to be online throughout the run of the protocol. Several other security properties which are of paramount importance are forward secrecy, resistance to collusion attacks, resistance to key impersonation attacks, etc. Our proposed scheme satisfies all these properties. A comparison of our protocol and other protocols is listed in the table below and this clearly highlights the salient features of the proposed scheme (Table 1).

Table 1. Comparison of certificateless key exchange protocols.

Protocol	Pairing-free	Reduced to CDH	Active adversary	Single round
Yang et al.	✓	✗	✗	✓
Sun et al.	✓	✗	✗	✓
Lippold et al.	✗	✓	✗	✓
Our scheme	✓	✓	✓	✓

2.2 Identity Based Cryptography

One of the strongest security models for identity based key agreement (IBKE) is the id-CK+ model proposed by Fujioka et al. [5] which is based on the CK model [1,10]. Since pairing is an extremely costly mathematical operation, we focus on schemes that do not involve the use of pairing. There are four schemes

in the literature by Fiore et al. [4], Gunther et al. [7], Saeednia et al. [14] and Sree Vivek et al. [17] which are pairing free and secure in this model. However, three of them are not secure in the presence of an active adversary as demonstrated in the paper by Sree Vivek et al. Other proposed schemes [2,9] were either broken subsequently or involve an initial agreement on who initiates the key agreement protocol. Therefore, we do not consider those schemes for our comparison. We propose an efficient pairing free scheme that is secure in this model and also in the presence of active adversaries. Additionally, our scheme is proven secure based on the hardness assumption of the Computational Diffie Hellman problem. Once again, several other security properties which our scheme satisfies are forward secrecy, resistance to collusion attacks, resistance to key impersonation attacks, etc. A comparison of our protocol and other protocols is listed in the table below and this clearly highlights the salient features of the proposed scheme. Our scheme can also be proven secure according to the CK and eCK models [11] which will be described in the full version of the paper. We observe that while the id-CK+ model is stronger than the CK model [5], the eCK and CK models are incomparable [3] (Table 2).

Table 2. Comparison of identity based key exchange protocols.

Protocol	Pairing-free	Reduced to CDH	Active adversary	Single round
Fiore et al.	✓	✓	✗	✓
Gunther et al.	✓	✗	✗	✗
Saeednia et al.	✓	✗	✗	✓
Sree Vivek et al.	✓	✗	✓	✓
Our scheme	✓	✓	✓	✓

3 A Certificateless Authenticated Key Exchange Protocol (CLAKE)

A certificateless key exchange protocol contains the following six probabilistic polynomial time algorithms - Setup, Partial Extract, Set Secret Value, Public Key Generation, Private Key Generation, Key Agreement.

Here, a particular user is denoted as U_A and his identity as ID_A. Additionally, we use the following naming scheme: UPK - User Public Key, FPK - Full Public Key, PPK - Partial Public Key, USK - User Secret Key, FSK - Full Secret Key, PSK - Partial Secret Key.

- **Setup (K):** This algorithm is run by the KGC. It generates the master secret key (MSK) first and then the public parameters (params), given a security parameter K as the input. The KGC publishes params and keeps the MSK secret.
- **Partial Extract (params, ID_A):** This algorithm is run by the KGC. Given params and user identity ID_A, this algorithm generates the Partial Secret Key (PSK) and the Partial Public Key (PPK) of a user U_A and sends them to the user. This can be sent over a public or private channel.

- **Set Secret Value (params, K, ID_A, PSK):** This algorithm is run by each user to generate his user secret key. The input to this algorithm is params, the security parameter K, the user's identity ID_A and the user's partial secret key PSK.
  The user secret key is not revealed to anyone.
- **Public Key Generation (params, ID_A, USK, PPK):** This algorithm is performed by the user. The input to this algorithm is params, the user identity ID_A corresponding to the user U_A, his user secret key and his partial public key. The output of this algorithm is the user generated public key. The full public key has two components - the partial public key together with the user public key.
- **Private Key Generation (params, ID_A, PSK, USK):** This algorithm is run by each user to generate his full private key. The input to this algorithm is params, the user identity ID_A corresponding to user U_A, his partial secret key and his user secret key. The output is his full secret key which is a tuple consisting of both the partial secret key and the user secret key. This is kept secret by the user and even KGC does not have full knowledge about it.
- **Key Agreement (params, ID_A, ID_B):** This algorithm is run by two users A and B who wish to compute a shared secret key. In order to do so, they take part in a session by exchanging components and eventually compute their shared secret which is unknown to other parties. The protocol could be initiated by either of the two users.

Key Sanity Check:
Key sanity check is done at two different places

- **User Verification:** Whenever the KGC gives the user a PPK and PSK, he runs a key sanity check to verify if the keys given by the KGC are valid.
- **Public Verification:** A different user ($\neq U_A$), who intends to use the public key of user U_A to take part in a key exchange protocol with A must first ensure that the public key he receives is valid. This consists of two checks - one for the partial public key and one for the user generated public key.

4 Security Models for CLAKE

There have been several security models proposed for certificateless key exchange protocols. The strongest model is the one introduced by Lippold et al. which is based on the Canetti-Krawczyk (CK) model for key agreement. In this paper, we define a new security model that is an improvisation of the Lippold et al. security model. The model considers an active adversary who can tamper with and replace messages going across the network. We propose a scheme that is pairing free, highly efficient and is secure in this model. Additionally, there are several other security features like forward secrecy, resistance to reflection attacks, security against collusion attack etc. Our scheme also satisfies these properties and this is discussed in more detail later on.

Let there be n parties in the network. The protocol may be run between any of these parties. Each run of the protocol is called as a session and the secret key computed in that run is called as the session key of the two parties involved. Each session can be initiated by either of the two parties involved and the user who initiates a session is called the initiator and the other user is called as the peer. $\pi_{i,j}^t$ represents the t^{th} session between parties i and j which is initiated by party i with intended partner party j. The session state of a user with identity ID_i taking part in a session is the set comprising of all the components he sends to the other user in that session.

For any certificateless crypto system, there are two types of adversaries A_I and A_{II}. A_I denotes a dishonest user who can replace other users' public keys but has no knowledge about the master secret key. A_{II} represents the malicious KGC who has knowledge of MSK but is trusted not to replace the public keys. However, in this model we also allow A_{II} to replace public keys.

The security game runs in two stages. During the first stage, the adversary is allowed to make the following queries in any order:

- **Hash Queries:** The adversary has access to all the hash oracles.
- **Reveal Partial Secret Key (ID_i):** The challenger responds with the partial secret key of user with identity ID_i.
- **Reveal User Secret Key (ID_i):** The challenger responds with the user generated secret key of the user with identity ID_i.
- **Replace Partial Public Key (ID_i, pk):** The challenger first checks that the given input pk is a valid partial public key for user with identity ID_i by running the user verification test. If it is indeed valid, party i's partial public key is replaced with pk chosen by the adversary. Party i will use the new partial public key for all communication and computation.
- **Replace User Generated Public Key (ID_i, pk):** The challenger first checks that the given input pk is a valid user generated public key for user with identity ID_i by running the public verification test. If it is indeed valid, party i's user generated public key is replaced with pk chosen by the adversary. Party i will use the new public key for all communication and computation.
- **Reveal Ephemeral Key ($\pi_{i,j}^t, i$):** The challenger responds with the ephemeral secret key used by party with identity ID_i in session $\pi_{i,j}^t$.
- **Session Simulation:** The adversary is allowed to ask shared secret key queries. The adversary queries for a shared secret belonging to a session established between two users i and j. The adversary can also emulate as one of the users, either i or j and present the challenger with the session state corresponding to that user. The challenger has to generate the session state for the other user of the session and obtain the shared secret key corresponding to that session. The adversary can also query for the session secret key between the two parties i and j from the challenger, where the adversary does not impersonate any of the users. In this case, the challenger has to generate the session state for both the users and obtain the shared secret key corresponding to that session and provide it to the adversary.

The key reveal queries can be classified into three categories:

- Reveal partial secret key: Which compromises the secret generated by the KGC and given to the user.
- Reveal user generated secret key: Which compromises the secret generated by the user as part of the full secret key.
- Reveal ephemeral secret key: Which compromises the transient secret generated by the user for that session alone.

A user is said to be fully corrupt with respect to a session if the adversary knows all the three secrets associated with that user for that session. At the end of the first stage, the adversary issues a test query as follows:

Test Session: The adversary randomly chooses a session $\pi^t_{A,B}$ between two users A and B for which it has not already queried the shared secret key and for which neither party is fully corrupted.

The challenger will toss a random bit $b \in_R \{0, 1\}$. If $b = 0$, the challenger will give the adversary the session key K_0 of the test session. If $b = 1$, the challenger will take a random shared secret key K_1 and give it to the adversary.

The adversary can continue to make queries as in the first phase, subject to certain restrictions which will be described later.

Guess: The adversary makes a guess b' as to which key K_0 or K_1 was given by the challenger. The adversary wins if $b' = b$. The certificateless key agreement protocol is said to be secure if no polynomial- time adversary has non-negligible advantage in winning the above game, i.e., distinguishing K_0 from K_1.

Note: There is no 'send' query present in this model as our protocol is single round and it is a 2-party protocol, thereby invalidating the need for it. Also, the adversary has access to the components exchanged and can modify them as per its wish as it is an active adversary.

4.1 Strong Type I Secure Certificateless Key Agreement Scheme

A certificateless key agreement scheme is Strong Type I secure if every probabilistic, polynomial-time adversary E has negligible advantage in winning the game described above subject to the following constraints:

- E may corrupt at most two out of three types of secrets per party involved in the test session.
- E is allowed to replace public keys of any party. However, this counts as the corruption of one secret. Replacing the partial public key and the user generated public key each correspond to the corruption of one secret.
- E may not ask to reveal the secret value of any identity for which it has replaced the corresponding public key. That is, E cannot ask to reveal the partial private key if it has already replaced the partial public key, and similarly cannot ask to reveal the user generated secret key if it has replaced the user generated public key.

- E is allowed to ask session key reveal queries even for session keys computed by identities where E has replaced either of the identities' public keys, but not both. Also, E is not allowed to ask for session keys where E has replaced the public keys of one party, and impersonates the other party generating its own ephemeral components.
- E may not replace the public keys of either of the identities that take part in the test query's session before the test query has been issued. However, it can replace their public keys after the test query subject to the fact that the test query's computation is done with respect to the unchanged public keys.
- E can tamper with any message that is exchanged between any two users in the system, i.e. the ephemeral components. However, E cannot ask for the ephemeral key of a user in a session where it has tampered with the components that the user sent. In other words, replacing the ephemeral components is also counted as corruption of one secret.

4.2 Strong Type II Secure Certificateless Key Agreement Scheme

A certificateless key agreement scheme is Strong Type II secure if every probabilistic, polynomial-time adversary E has negligible advantage in winning the game described above subject to the following constraints:

- E is given the master secret key at the start of the game. Therefore, E has knowledge of the partial secret keys of all the users in the network.
- The rest of the properties are same as a strong type I adversary (from the second point onwards)

4.3 Why is this Model an Extension?

The model we have defined in this paper is an extension of the Lippold et al. model because we allow the adversary to replace both the partial public keys and user generated public keys. Furthermore, we give the adversary the freedom to replace either of the two alone and not necessarily both, and possibly have a chance to get the other secret. For example, the adversary could replace the partial public key and ask for the user generated secret key of a user. In the Lippold et al. model, the adversary was only given the power to replace the user generated public key and not the partial public key generated by the KGC (such a notion was not present in the model). Also, in our scheme, we provide a sanity check which helps a user to determine whether the ephemeral messages he received were infact sent by the intended party or were modified by an active adversary. Note that in a single round protocol only sanity checks and error detection are possible and not error correction if the adversary tampered with the message.

5 CLAKE Scheme

- **Setup (K):** Given K as security parameter, the key generating center (KGC) chooses a group $\mathbb{G}$ of order p and generator of this group P. Then, x is chosen

randomly from $\mathbb{Z}_q^*$. KGC sets the master secret key (MSK) as x and sets the master public key as xP. The KGC chooses 5 hash functions defined below:

- $\mathbb{H}_1$: $\{0,1\}^* \times \mathbb{G} \times \rightarrow \mathbb{Z}_q^*$
- $\mathbb{H}_2$: $\{0,1\}^* \times \{0,1\}^* \times \mathbb{G}^5 \rightarrow \mathbb{G}$
- $\mathbb{H}_3$: $\{0,1\}^* \times \mathbb{G} \times \mathbb{G} \rightarrow \mathbb{Z}_q^*$
- $\mathbb{H}_4$: $\{0,1\}^* \times \mathbb{G} \times \mathbb{G} \rightarrow \mathbb{Z}_q^*$
- $\mathbb{H}_5$: $\{0,1\}^* \times \mathbb{G} \times \mathbb{G} \rightarrow \mathbb{Z}_q^*$

KGC keeps MSK secret and makes params public, where params = (K,xP,$\mathbb{H}_1$, $\mathbb{H}_2$,$\mathbb{H}_3$,$\mathbb{H}_4$,$\mathbb{H}_5$).

Note: We use the following naming scheme:
UPK - User Public Key, FPK - Full Public Key, PPK - Partial Public Key, USK - User Secret Key, FSK - Full Secret Key, PSK - Partial Secret Key.

– **Partial Extract** (params, ID_i)**:** Given an identity ID_i, the KGC does the following to generate the partial public key (PPK) and the partial secret key (PSK).
  - Choose randomly $r_i \in_R \mathbb{Z}_q^*$. Compute $R_i = rP$
  - Compute $h_i = H_1(ID_i, R_i)$ and $s_i = r_i + xh_i$

  Return PSK = $< s_i >$ and PPK = $< R_i, s_iP >$.

  Key Sanity Check by User

  Now, the user can verify whether the partial keys received from the KGC were valid using the following check:
  - $s_iP = R_i + H_1(ID_i, R_i)xP$

  If the equality is satisfied, the keys given by the KGC are valid.

– **User Secret Key** (params, ID_i, PSK)**:** After receiving the partial keys from the KGC, a user with identity ID_i does the following to generate the user secret key (USK) and user public key (UPK).
  - Choose randomly $y_i \in_R \mathbb{Z}_q^*$
  - Compute $x_i = y_i + s_iH_1(ID_i, y_iP)$

  Set USK = $< x_i >$ and UPK = $< x_iP, y_iP >$.

– **Full Private Key** (params, ID_i, PSK, USK)**:** The user with identity ID_i runs this algorithm and sets his full private key FSK as $< x_i, s_i >$.

– **Full Public Key** (params, ID_i, PSK, PPK, USK, UPK)**:** The user with identity ID_i runs this algorithm and sets his full public key FPK as $< x_iP, y_iP, s_iP, R_i >$.

  Key Sanity Check For Public Verification

  Anyone who intends to use the public key of a user with identity ID_i must first ensure that the available public key is valid. This can be done by the following two checks:
  - $s_iP = R_i + H_1(ID_i, R_i)xP$
  - $x_iP = y_iP + H_1(ID_i, y_iP)s_iP$

  If both the equalities are satisfied, the available public key is valid.

– **Key Agreement**

  Two users A and B with identities ID_A and ID_B who wish to agree upon a shared secret key choose ephemeral secrets respectively and engage in a session as described below. As it is a single round protocol, without loss of generality, let's assume that the session is initiated by A.

User A: Chooses his ephemeral components as follows:
- Choose $z_A \in_R \mathbb{Z}_q^*$ and compute $t_A = z_A + x_A H_1(ID_A, z_A P)$

A sets his ephemeral key as t_A.
Then, A sends the following to B: $< ID_A, t_A P, z_A P >$.
User B: First verifies that the components he received from A were valid using the following check:

$$t_A P = z_A P + H_1(ID_A, z_A P) x_A P$$

If the equality is satisfied, the components sent by A are valid. This helps to detect whether an active adversary tampered with the message. Now, user B chooses his ephemeral components as follows:
- Choose $z_B \in_R \mathbb{Z}_q^*$ and compute $t_B = z_B + x_B H_1(ID_B, z_B P)$

B sets his ephemeral key as t_B.
Then, B sends the following to A: $< ID_B, t_B P, z_B P >$.
Shared Secret Computation
- **User A:** First verifies that the components he received from B were valid using the following check:

  $$t_B P = z_B P + H_1(ID_B, z_B P) x_B P$$

  If the equality is satisfied, the components sent by B are valid. Then, A does the following to compute the shared secret:

  * $K_1 = \{s_A + t_A H_3(ID_A, s_A P, t_A P)\}\ \{s_B P + H_3(ID_B, s_B P, t_B P) t_B P\}$
    $K_2 = \{x_A + t_A H_4(ID_A, x_A P, t_A P)\}\ \{x_B P + H_4(ID_B, x_B P, t_B P) t_B P\}$
    $K_3 = \{s_A + x_A H_5(ID_A, s_A P, x_A P)\}\ \{s_B P + H_5(ID_B, s_B P, x_B P) x_B P\}$
    $SK = H_2(ID_A, ID_B, t_A P, t_B P, K_1, K_2, K_3)$
    The shared secret is SK.

- **User B:** Does the following to compute the shared secret:

  * $K_1 = \{s_A P + H_3(ID_A, s_A P, t_A P) t_A P\}\ \{s_B + H_3(ID_B, s_B P, t_B P) t_B\}$
    $K_2 = \{x_A P + H_4(ID_A, x_A P, t_A P) t_A P\}\ \{x_B + H_4(ID_B, x_B P, t_B P) t_B\}$
    $K_3 = \{s_A P + H_5(ID_A, s_A P, x_A P) x_A P\}\ \{s_B + H_5(ID_B, s_B P, x_B P) x_B\}$
    $SK = H_2(ID_A, ID_B, t_A P, t_B P, K_1, K_2, K_3)$
    The shared secret is SK.

  It can be observed that the shared secret computed by both of them is the same.

6 Security Proof

In the following proof, all the hash functions are modeled as random oracles. Here is a brief intuition behind the security proof of the scheme. Observe that there are totally six secret components for the parties A and B taking part in the test session. They are s_A, x_A, t_A corresponding to the secrets of party A and s_B, x_B, t_B corresponding to the secrets of party B. The adversary can access at

most four of the above six components and not more than two out of the three secrets per party. As a result, we will inject the hard problem instance in the other two components which are not revealed to the adversary. This explains the necessity for the three equations K_1, K_2 and K_3 in the key agreement as each of them contain a few components that would help to compute the solution to the hard problem depending on which of the secrets the adversary has queried. In other words, in some situations we would use K_1 to compute the solution to the hard problem and in other cases K_2 or K_3 depending on the queries made by the adversary.

6.1 Proof for Type I Adversary

Theorem 1. If there exists an adversary A_I that can break the above scheme with probability ϵ in time t_{adv}, then there exists a challenger C who can solve the CDH problem with probability atleast ϵ' in time t_{ch}, such that

$$\epsilon' \geq \epsilon\{(1/9t * q_{h_1}^2)(1 - \frac{1}{q})(1 - \frac{4}{q_{pkr}})(1 - \frac{2}{q_{ekq}})\}$$

and ϵ' is a non-negligible quantity if ϵ is non-negligible.

$$t_{ch} = S + t_{adv} + (q_1 + q_2 + q_3 + q_4 + q_5 + q_{ekq} + q_{psq} + q_{usq} + q_{fpq} + q_{sq} + q_{pkr})O(1)$$

which is polynomial if the time taken by the adversary is polynomial.
q_{id} = number of distinct identities queried by the adversary, q = order of the group $\mathbb{G}$ in which the hard problem can be solved by adversary to break the system. q_i = number of queries to the H_i hash oracle (where $i = 1, 2..5$). q_{ekq} = number of ephemeral key queries, q_{psq} = number of partial extract queries, q_{usq} = number of user secret key queries, q_{fpq} = number of full public key queries, q_{sq} = number of simulation queries, q_{pkr} = number of public key replacements made and S represents the time taken for the calculations performed by the challenger after the adversary returns his guess.

Proof. Let C be given an instance of the CDH problem (P, aP, bP). Suppose there exists a type I adversary, who is capable of breaking the key agreement scheme above, then C's aim is to find the value of abP.

Setup: The challenger C must set up the system exactly as given in the scheme. C chooses a random number $x \in Z_q^*$ and sets the MSK as x and the master public key as xP. The master public key is given to the adversary while the master secret key is not revealed. C then chooses five hash functions, $\mathbb{H}_i$, where $i = 1, 2..5$ and models them as random oracles. Also C maintains a list l_i for each hash function to maintain consistency. C also maintains a list l_{id} for storing all the keys. Each entry of l_{id} is of the form $< ID, FPK, PSK, USK, FSK, PPK, UPK, X_i, Y_i >$, where the bits X_i and Y_i are used to determine whether the partial and user generated public keys have been replaced or not.

Training Phase: The adversary A_1 makes use of all the oracles provided by C. The system is simulated in such a way that A_1 cannot differentiate between a real and a simulated system that is provided by C.

Choosing the Target Identities: In the oracle $O_{\mathbb{H}_1(ID_i,(R_j))}$, the adversary asks q_{h_1} queries and expects a response from the challenger for each of them. Since the adversary can query on the same ID and different R_j's, the number of distinct identities queried is different from q_{h_1}. Let that number be q_{id}. $1 \leq q_{id} \leq q_{h_1}$. The challenger randomly chooses two queries with different identities ID_A and ID_B sets the target identities to be those. Also, the challenger chooses a random number t such that $1 \leq t \leq q_{h_1}$ and sets the test session to be $\pi^t_{A,B}$. There are six secrets corresponding to the identities taking part in the test session. They are:
s_A, x_A, t_A which are the partial secret key, user secret key and ephemeral secret key of A respectively and s_B, x_B, t_B which are the partial secret key, user secret key and ephemeral secret key of B respectively.

Case 1: The adversary doesn't know the ephemeral keys t_A and t_B of the test session.

- **Oracle** $O_{\mathbb{H}_1}(ID_i, R_i)$:
  A list l_{h_1} is maintained of the form $< ID_i, R_i, h_i >$. C responds as follows:
  - If $< ID_i, R_i, h_i >$ already exists in the list then respond with value h_i from the list.
  - Else, choose a $h_i \in_R \mathbb{Z}^*_q$. Return h_i and add the tuple, $< ID_i, R_i, h_i >$ to the list.

  The response to the other hash oracles is similar to the first one and is not described here.
- **Oracle Partial Extract:** C responds as follows:
  - If values corresponding to ID_i already exists on the list l_{id}, then return s_i as PSK and (R_i, s_iP) as PPK from the list
  - Else,
    Choose $r_i \in_R \mathbb{Z}^*_q$. Compute $R_i = r_iP$
    Compute $h_i = H_1(ID_i, R_i)$ and $s_i = r_i + xh_i$.
    Output $< s_i >$ as the PSK and $< s_iP, R_i >$ as PPK. Add these values to the list l_{id} in the entry corresponding to ID_i.

Lemma 1. The above oracle outputs valid PSK and PPK.
Proof. It can be seen that the outputs given by the oracle satisfy the condition for a valid PPK, PSK.

- **Oracle User Private Key:** Challenger responds as follows:
  - If values corresponding to ID_i already exists on the list, then return $< s_i, x_i >$ from the list.
  - Else, if s_i is already in the list l_{id}, in the entry corresponding to ID_i, retrieve them.
    Else run the partial key extract oracle and retrieve that value.
    Choose $y_i \in_R \mathbb{Z}^*_q$
    Compute $h_i = H_1(ID_i, y_iP)$ and $x_i = y_i + s_ih_i$.
    Output $< x_i >$ as the user generated private key and add it to the list l_{id}.
    The corresponding user public key is $< x_iP, y_iP >$.

- **Oracle Public Key Generation:** Challenger responds as follows:
  - If values corresponding to ID_i already exists on the list, then return $< x_iP, y_iP, R_i, s_iP >$ from it.
  - Else, if (R_i, s_iP) are already in the list l_{id}, in the entry corresponding to ID_i, retrieve them. Else run the partial key extract oracle and retrieve those values.
    If (y_iP, x_iP) are already in the list l_{id}, in the entry corresponding to ID_i, retrieve them. Else run the user private key extract oracle and retrieve those values.
    Output (R_i, s_iP, y_iP, x_iP) as the full public key. Add these values to the list l_{id} in the entry corresponding to ID_i and set $X_i = 0, Y_i = 0$.

Lemma 2. The above oracle for public key generation outputs a valid full public key.

Proof. It can be observed that the output generated by the oracle passes the key sanity check for public verification mentioned in the scheme.

- **Oracle Partial Public Key Replace:** If the adversary tries to replace the partial public key for the identities taking part in the key exchange before the test query has been issued, the challenger will abort. Else, the adversary sends the values $< ID, R_i, s_iP >$ to the challenger C. The challenger runs the key sanity check for verifying the partial public key. If the test succeeds it adds these values to the list in the entry corresponding to ID and sets $X_i = 1$ to indicate that the partial public key has been replaced. Further key exchanges for this identity use this value of the partial public key.
- **Oracle User Generated Public Key Replace:** If the adversary tries to replace the user generated public key for the identities taking part in the key exchange before the test query has been issued, the challenger will abort. Else, the adversary sends the values $< ID, y_iP, x_iP >$ to the challenger C. The challenger runs the public key verification test. If the test succeeds it adds these values to the list in the entry corresponding to ID and sets $Y_i = 1$ to indicate that the user public key has been replaced. Further key exchanges for this identity use this value of the user public key.
- **Oracle Reveal Ephemeral Key:** Challenger responds as follows:
  - If the adversary asks to reveal the ephemeral key for the identities taking part in the key exchange for the session corresponding to the test session, the challenger will abort.
  - If values corresponding to ID_i for the session π_{ij}^t already exists, then return $< t_i >$.
  - Else, if x_i is already in the list l_{id}, in the entry corresponding to ID_i, retrieve them.
    Else run the user private key oracle and retrieve that value.
    Choose $z_i \in_R \mathbb{Z}_q^*$
    Compute $h_i = H_1(ID_i, z_iP)$ and $t_i = z_i + x_ih_i$.
    Output $< t_i >$ as the ephemeral key and store that value.

– **Session Simulation:**
The adversary asks for the shared secret between two users i and j for a session t. The adversary can also act as one of the users and present the session state of that user and ask the challenger to generate the session state of the other user and compute the shared secret key.
Case 1: The adversary does not act as either of the users.
The challenger generates the ephemeral components of both the parties and gives the following to the adversary: The session state of i as (ID_i, T_i, z_iP) and the session state of j as (ID_j, T_j, z_jP). Now, the adversary could have corrupted two out of the three secrets of both the parties i and j. Also, the adversary could have replaced the public keys of either user. Suppose it was for user j. The challenger computes the shared secret sk the same way user i would since he knows the secret keys of i. The challenger returns sk to the adversary as the shared secret. Similarly, if the adversary had replaced i's public keys, the challenger would have computed sk the same way j would have. The other cases where the adversary didn't replace the public keys of either party but corrupted the parties by just learning the secrets are easily covered as the challenger can compute the secret key the same way as either party would. Also, cases where the adversary replaced only one of the two possible public keys of one user are weaker cases than the above and can be easily handled.
Case 2: The adversary acts as user j and sends the session state to the challenger. The challenger generates the ephemeral components of user i and gives the following to the adversary: The session state of i as (ID_i, T_i, z_iP). Here, the challenger may or may not know the ephemeral secret key of j. The adversary could have corrupted two out of the three secrets of both the parties i and j. Also, the adversary could have replaced the public keys of either user. Suppose it was for user j. Then, the challenger computes the shared secret sk the same way user i would as he knows the secret keys of i.

The challenger returns sk to the adversary as the shared secret. If the adversary had replaced the public keys of user i, then the challenger aborts as this is not allowed as per the security model described earlier.

– **Test Session:**
The adversary gives the following session id $\pi_{i,j}^t$ to the challenger. Since the adversary knows 4 of the secrets s_A, x_A, s_B, x_B, the challenger injects the hard problem instance in the ephemeral components in the following way:
- Compute $t_AP = aP$, $t_BP = bP$, implicitly setting $t_A = a, t_B = b$
- Choose two random values c, d
- Compute $z_AP = t_AP - cx_AP$,$z_BP = t_BP - dx_BP$
- Set $H_1(ID_A, z_AP) = c$ and $H_1(ID_B, z_BP) = d$

The challenger sends the adversary the session state of A as (ID_A, t_AP, z_AP) and the session state of B as (ID_B, t_BP, z_BP).

Next, the challenger chooses a random group element Z and sends that to the adversary as the shared secret. This won't be a valid shared secret key. So, if the adversary breaks the scheme, he would guess that this isn't a valid shared secret key and return the bit 1. But in order to find that

this is invalid the adversary should have queried the H_2 oracle with a valid tuple $(ID_A, ID_B, t_AP, t_BP, k^1_{AB}, k^2_{AB}, k^3_{AB})$. Using this query, the challenger can solve the CDH problem by computing $S = k^1_{AB} - (s_A)(S_BP) - s_A(t_BP)H_3(ID_B, s_BP, t_BP) - (s_B)(t_AP)H_3(ID_A, s_AP, t_AP)$.
The challenger returns S as the solution to the hard problem.

– **Correctness:**
- We know that $k^1_{AB} = (s_A + t_AH_3(ID_A, s_AP, t_AP))\ (s_BP + t_BPH_3(ID_B, s_BP, t_BP))$
- This shows that $S = t_At_BP$
- Since $t_A = a$, $t_B = b$ implicitly, S is the solution to the CDH problem.

Probability Analysis:

The challenger fails only if any of the following events occur:
- E_1 : The test session chosen by the adversary is not the same as the one chosen by the challenger.
- E_2 : An invalid public key replacement by the adversary was not detected.
- E_3 : The adversary tried to replace the partial public key or the user generated public key for one of the identities in the test session.
- E_4 : The adversary asked to reveal the ephemeral key for one of the identities in the test session for the session corresponding to the test session.

Let t be the maximum number of sessions between any two parties.

$$Pr[E_1] = (1 - 1/(t * q^2_{h_1})); Pr[E_2] = \left(\frac{1}{q}\right)$$
$$Pr[E_3] = \left(\frac{4}{q_{pkr}}\right); Pr[E_4] = \left(\frac{2}{q_{ekq}}\right)$$

Therefore, the probability of the challenger being successful is atleast $Pr[\neg(E_1 \vee E_2 \vee E_3 \vee E_4)]$. And the advantage of the adversary is ϵ. Also,there are 9 possible cases that could happen with equal probability. Thus,

$$\epsilon' \geq \epsilon\{(1/9t * q^2_{h_1})(1 - \frac{1}{q})(1 - \frac{4}{q_{pkr}})(1 - \frac{2}{q_{ekq}})\}$$

and ϵ' is non-negligible whenever ϵ is non-negligible. Also, it can be easily seen that $t_{ch} = S + t_{adv} + (q_1 + q_2 + q_3 + q_4 + q_5 + q_{ekq} + q_{psq} + q_{usq} + q_{fpq} + q_{sq} + q_{pkr})O(1)$.

The other 8 cases are described in the below table (Table 3):

6.2 Proof for Type II Adversary

The proof is very similar to the proof in the case of the type I adversary and will be described in the full version of the paper.

Table 3. Security proof for Type 1 adversary.

Case	Unknown to Adv A_I	Hard problem instance
2	t_A, s_B	$t_A P = aP$, $s_B P = bP$
3	t_A, x_B	$t_A P = aP$, $x_B P = bP$
4	s_A, s_B	$s_A P = aP$, $s_B P = bP$
5	s_A, x_B	$s_A P = aP$, $x_B P = bP$
6	s_A, t_B	$s_A P = aP$, $t_B P = bP$
7	x_A, s_B	$x_A P = aP$, $s_B P = bP$
8	x_A, x_B	$x_A P = aP$, $x_B P = bP$
9	x_A, t_B	$x_A P = aP$, $t_B P = bP$

7 Additional Security Properties

Our proposed CLAKE scheme satisfies several additional security properties.

- **Strong Forward Secrecy:** Learning the private keys of parties should not affect the security of the shared secret key.
- **Resistance to Reflection Attacks:** Both parties in the session have the same identity.
- **Resistance to Collusion Attack:** Several users should not be able to collude and compute the secret keys of some other user.
- **Resistance to Key Compromise Impersonation Attacks:** The knowledge of a user's full private key should not allow the adversary to impersonate another party to that user.
- **Resistance to Ephemeral Key Compromise Impersonation Attacks:** The knowledge of a user's ephemeral key in one session should not allow an adversary to impersonate another party to that user.
- **Known Session Key Security:** A compromised session key does not compromise past or future sessions.
- **Unknown Key Share:** A user A cannot be coerced into sharing a key with C when infact A thinks he is sharing a key with B.

A detailed proof of security for all these properties will be described in the full version of the paper.

8 Identity based Key Exchange Protocol (IBKE)

In IBKE protocols, the KGC maintains the master public key and master secret key and generates the private key s_i for each user. An identity based key exchange protocol contains the following three probabilistic polynomial time algorithms - Setup, Key Generation, Key Agreement.

Here, a particular user is denoted as U_A and his identity as ID_A. Additionally, we use the following naming scheme: UPK - User Public Key. USK - User Private Key.

- **Setup (K):** This algorithm is run by the KGC. It generates the master secret key (MSK) first and then the public parameters (params), given a security parameter K as the input. Along with the other information, params additionally contains α. The KGC publishes params and keeps the MSK secret.
- **Key Generation (params, ID_A):** This algorithm is run by the KGC. Given params and user identity ID_A, this algorithm generates the private key of the user (USK) and the corresponding public Key (UPK) and sends them to the user. This can be sent over a public or private channel.
- **Key Agreement (params, ID_A, ID_B):** This algorithm is run by two users A and B who wish to compute a shared secret key. In order to do so, they take part in a session by exchanging components and eventually compute their shared secret which is unknown to other parties. The protocol could be initiated by either of the two users.

9 Security Model for IBKE

There have been several security models proposed for identity based key exchange protocols. We follow the id-CK+ model used by Fujioko et al. which is based on the Canetti-Krawczyk (CK) model for key agreement. We propose a scheme that is pairing free, highly efficient and is secure in this model. Additionally, there are several security features that are still not covered in the model like forward secrecy, resistance to reflection attacks, security against collusion attack etc. Our scheme also satisfies these properties and this is discussed in more detail later on.

We consider an adversary who is given access to the private keys of polynomial number of users. It can also impersonate as any other user. This is the strongest adversary and we prove our scheme secure against this type of adversary. The setting with n parties and the way they can exchange messages is same as in CLAKE. The security game runs in two stages. During the first stage, the adversary is allowed to make the following queries in any order:

- **Hash Queries:** The adversary has access to all the hash oracles.
- **Party Corruption (ID_i):** The challenger responds with the private key of the user with identity ID_i.
- **Reveal Ephemeral Key ($\pi_{i,j}^t, i$):** The challenger responds with the ephemeral secret key used by party with identity ID_i in session $\pi_{i,j}^t$.
- **Session Simulation:** Same as in the security model for CLAKE.

A party is said to be fully corrupted with respect to a session if the adversary knows both the private key and the ephemeral secret key. At the end of the first stage, the adversary issues a test query as follows:

Test Session: This is same as in the test session in the security model for CLAKE.

10 Identity based Scheme

– **Setup (K):** Given K as security parameter, the key generating center (KGC) chooses a group $\mathbb{G}$ of order p and generator of this group P. Then x is chosen randomly from $\mathbb{Z}_q^*$. The KGC sets the master secret key (MSK) as x and sets the master public key as xP. The KGC chooses 3 hash functions defined below:
  - $\mathbb{H}_1$: $\{0,1\}^* \times \mathbb{G}\times \rightarrow \mathbb{Z}_q^*$
  - $\mathbb{H}_2$: $\{0,1\}^* \times \{0,1\}^* \times \mathbb{G}^5 \rightarrow \mathbb{G}$
  - $\mathbb{H}_3$: $\{0,1\}^* \times \mathbb{G} \times \mathbb{G} \rightarrow \mathbb{Z}_q^*$

  The KGC keeps the MSK secret and makes params public, where params = (K,xP, $\mathbb{H}_1$,$\mathbb{H}_2$,$\mathbb{H}_3$,$\mathbb{H}_4$,$\mathbb{H}_5$).
  Note: We use the following naming scheme:
  UPK - User Public Key, USK - User Private Key
– **Key Generation** (params, ID_i)**:** Given an identity ID_i, the KGC does the following to generate the public key (UPK) and the private key (USK) of the user.
  - Choose randomly $r_i \in_R \mathbb{Z}_q^*$. Compute $R_i = rP$
  - Compute $h_i = H_1(ID_i, R_i)$, $s_i = r_i + xh_i$
  - Return USK = $< s_i >$ and UPK = R_i, s_iP.

  Key Sanity Check by User
  Same as in the CLAKE scheme where the user verifies the partial keys received from the KGC.
– **Key Agreement**
  The two users A and B with identities ID_A and ID_B who wish to agree upon a shared secret key choose ephemeral secret components respectively and then engage in a session as described below. Without loss of generality, let's assume that the session is initiated by A.
  User A: Chooses his ephemeral components as follows:
  - Choose $z_A \in_R \mathbb{Z}_q^*$ and compute $t_A = z_A + s_A H_1(ID_A, z_AP)$
    A sets his ephemeral key as t_A. Then, A sends the following to B: $< ID_A, t_AP, z_AP >$

  User B: First verifies that the components he received from A were valid using the following check:

$$t_AP = z_AP + H_1(ID_A, z_AP)s_AP$$

  If the equality is satisfied, the components sent by A are valid. Now, user B chooses his ephemeral components as follows:
  - Choose $z_B \in_R \mathbb{Z}_q^*$ and compute $t_B = z_B + s_B H_1(ID_B, z_BP)$
    B sets his ephemeral key as t_B. Then, B sends the following to A: $< ID_B, t_BP, z_BP >$

  Shared Secret Computation
  - **User A:** First verifies that the components he received from B were valid using the following check:

$$t_BP = z_BP + H_1(ID_B, z_BP)s_BP$$

If the equality is satisfied, the components sent by B are valid. User A does the following to compute the shared secret:

$* K_1 = \{s_A + t_A H_3(ID_A, s_A P, t_A P)\} \{s_B P + H_3(ID_B, s_B P, t_B P) t_B P\}$
$SK = H_2(ID_A, ID_B, t_A P, t_B P, K_1)$
The shared secret is SK.

- **User B:** Does the following to compute the shared secret:

$* K_1 = \{s_A P + H_3(ID_A, s_A P, t_A P) t_A P\} \{s_B + H_3(ID_B, s_B P, t_B P) t_B\}$
$SK = H_2(ID_A, ID_B, t_A P, t_B P, K_1)$
The shared secret is SK.

The shared secret computed by both of them is the same.

11 Security Proof for IBKE

In the following proof, all the hash functions are modeled as random oracles.

Theorem 1. If there exists an adversary E that can break the above scheme with probability ϵ in time t_{adv}, then there exists a challenger C who can solve the CDH problem with probability atleast ϵ' in time t_{ch}, such that

$$\epsilon' \geq \epsilon\{(1/(4t * q_{h_1}^2))(1 - \frac{2}{q_{ekq}})\}$$

and ϵ' is a non-negligible quantity if ϵ is non-negligible.
$t_{ch} = S + t_{adv} + (q_1 + q_2 + q_3 + q_{ekq} + q_{usq} + q_{sq})O(1)$ which is polynomial if the time taken by the adversary is polynomial.
q_{id} = number of distinct identities queried by the adversary, q = order of the group $\mathbb{G}$ in which the hard problem can be solved by adversary to break the system.
q_i = number of queries to the H_i hash oracle (where $i = 1, 2, 3$).
q_{ekq} = number of ephemeral key queries, q_{usq} = number of user secret key queries, q_{sq} = number of simulation queries and S represents the time taken for the calculations performed by the challenger after the adversary returns his guess.

Proof. Let C be given an instance of the CDH problem (P, aP, bP). Suppose there exists an adversary, who is capable of breaking the key agreement scheme above, then C's aim is to find the value of abP.

Setup: The challenger C must set up the system exactly as given in the scheme. C chooses a random number $x \in Z_q^*$ and sets the MSK as x and the master public key as xP. The master public key is given to the adversary while the master secret key is not revealed. C then chooses three hash functions, $\mathbb{H}_i$, where $i = 1, 2, 3$ and models them as random oracles. Also C maintains a list l_i for each hash function to maintain consistency. C also maintains a list l_{id} for storing all the keys. Each entry of the l_{id} is of the form, $< ID, USK, UPK >$.

Training Phase: In this phase the adversary E makes use of all the oracles provided by C. The system is simulated in such a way that E cannot differentiate between a real and a simulated system that is provided by C.

Choosing the Target Identities: The target identities are chosen the same way as in the CLAKE security proof. Now there are four secrets corresponding to the identities taking part in the test session. They are: s_A, t_A which are the private key and ephemeral secret key of A respectively and s_B, t_B which are the private key and ephemeral secret key of B respectively.

Case 1: The adversary doesn't know the ephemeral keys t_A and t_B of the test session.

- **Oracle** $O_{\mathbb{H}_1}$ (ID_i, R_i)**:**
  The response to the hash oracles is same as in the CLAKE proof and is not described here.
- **Oracle Reveal Private Key:** The challenger's response is as follows:
  - If values corresponding to ID_i already exists on the list l_{id}, then return s_i as USK from the list
  - Else,
    Choose $r_i \in_R \mathbb{Z}_q^*$.
    Compute $R_i = r_i P$
    Compute $h_i = H_1(ID_i, R_i)$ by querying the H_1 oracle.
    Compute $s_i = r_i + xh_i$. Output $< s_i >$ as the USK and set $< s_i P, R_i >$ as UPK. Add these values to the list l_{id} in the entry corresponding to ID_i.

Lemma 1. The above oracle outputs valid USK and UPK.
Proof. It can be observed that the outputs given by the oracle satisfy the condition for a valid USK, UPK. (They satisfy the key sanity check for user verification given earlier).

- **Oracle Public Key Generation:** The challenger's response is as follows:
  - If values corresponding to ID_i already exists on the list, then return $< R_i, s_i P >$ from the list.
  - Else,
    Run the private key extract oracle and retrieve those values. Output $< R_i, s_i P >$ as the full public key.
- **Oracle Reveal Ephemeral Key:** The challenger's response is as follows:
  - If the adversary asks to reveal the ephemeral key for the identities ID_A or ID_B for the session corresponding to the test session, the challenger will abort.
  - If values corresponding to ID_i for the session π_{ij}^t already exists, then return $< t_i >$.
  - Else,
    If s_i is already in the list l_{id}, in the entry corresponding to ID_i, retrieve them. Else run the private key oracle and retrieve that value.
    Choose $z_i \in_R \mathbb{Z}_q^*$

Compute $h_i = H_1(ID_i, z_iP)$ by querying the H_1 oracle.
Compute $t_i = z_i + s_ih_i$.
Output $< t_i >$ as the ephemeral key and store the value.

– **Session Simulation:** The adversary asks for the shared secret between two users i and j for a session t. The adversary can also act as one of the users and present the session state of that user and ask the challenger to generate the session state of the other user and compute the shared secret key.
Case 1: The adversary does not act as either of the users.
The challenger generates the ephemeral components of both the parties and gives the following to the adversary: The session state of i as (ID_i, T_i, z_iP) and the session state of j as (ID_j, T_j, z_jP). The challenger computes the shared secret sk the same way user i would since he knows the secret keys of i. The challenger returns sk to the adversary as the shared secret.
Case 2: The adversary acts as user j and sends the session state to the challenger.
The challenger generates the ephemeral components of user i and gives the following to the adversary: The session state of i as (ID_i, T_i, z_iP). Here, the challenger may or may not know the ephemeral secret key of j. The challenger computes the shared secret sk the same way user i would since he knows the secret keys of i. The challenger returns sk to the adversary as the shared secret.

– **Test Session:**
The adversary gives the following session id $\pi^t_{i,j}$ to the challenger.
Since the adversary knows 2 of the secrets (s_A, s_B), the challenger injects the hard problem instance in the ephemeral components in the following way:
- Set $t_A = a$, $t_B = b$
- Compute $t_AP = aP$, $t_BP = bP$
- Choose two random values c, d
- Compute $z_AP = t_AP - cs_AP$
- Compute $z_BP = t_BP - ds_BP$
- Set $H_1(ID_A, z_AP) = c$
- Set $H_1(ID_B, z_BP) = d$

The challenger sends the adversary the session state of A as (ID_A, t_AP, z_AP) and the session state of B as (ID_B, t_BP, z_BP).
Next, the challenger chooses a random group element Z and sends that to the adversary as the shared secret. This won't be a valid shared secret key. So, if the adversary breaks the scheme, he would guess that this isn't a valid shared secret key and return the bit 1. But in order to find that this is invalid the adversary must have queried the H_2 oracle with a valid tuple $(ID_A, ID_B, t_AP, t_BP, k_1)$. Using this query, the challenger can solve the CDH problem.
It computes $S = \{H_3(ID_A, s_AP, t_AP)H_3(ID_B, s_BP, t_BP)\}^{-1}\{k_1 - (s_A)(S_BP) - s_A(t_BP)H_3(ID_B, s_BP, t_BP) - (s_B)(t_AP)H_3(ID_A, s_AP, t_AP)\}$.
The challenger returns S as the solution to the hard problem.

– **Correctness:**
  - We know that $k_1 = (s_A + t_A H_3(ID_A, s_A P, t_A P)(s_B P + t_B P H_3(ID_B, s_B P, t_B P)$
  - This shows that $S = t_A t_B P$
  - Since $t_A = a$, $t_B = b$, S is the solution to the CDH problem.

  Probability Analysis: The probability analysis is similar to (and simpler than) the one given in the proof for the CLAKE scheme and hence is not described here.

The other 3 cases are described in the below table (Table 4):

Table 4. Security proof.

Case	Unknown to Adv A_I	Hard problem instance
2	t_A, s_B	$t_A P = aP$, $s_B P = bP$
3	s_A, s_B	$s_A P = aP$, $s_B P = bP$
4	s_A, t_B	$s_A P = aP$, $t_B P = bP$

Our proposed identity based key agreement scheme satisfies all the additional security properties described in Sect. 7. The proof of security is similar to that in the CLAKE scheme.

12 Conclusions

In this paper, we propose a security model for certificateless key exchange protocols that is an extension of previously existing models. We note that previously existing pairing-free protocols are not secure in this model and we design a highly efficient pairing-free certificateless authenticated key exchange protocol that is secure in this model. Our scheme also has the advantages of having a single round of communication between the pair of users and there is no predefined order in which messages are exchanged between the users. Also, our scheme is the first pairing-free certificateless key exchange protocol secure based on the CDH assumption. The previously existing schemes were secure based on much stronger assumptions like the Gap-Diffie Hellman assumption. Finally, we use a restriction of our scheme to design an efficient pairing-free identity based key agreement protocol that is secure in the id-CK+ security model and we prove its security based on the hardness assumption of the CDH problem. Our identity based scheme is also a single round protocol. Additionally, both our schemes satisfy several other security properties such as resistance to collusion attacks, forward secrecy etc. We prove the security of both our schemes in the random oracle model. An open problem is to design schemes satisfying all these properties that is proven secure in the standard model.

References

1. Canetti, R., Krawczyk, H.: Analysis of key-exchange protocols and their use for building secure channels. In: Pfitzmann, B. (ed.) EUROCRYPT 2001. LNCS, vol. 2045, pp. 453–474. Springer, Heidelberg (2001)
2. Cao, X., Kou, W., Du, X.: A pairing-free identity-based authenticated key agreement protocol with minimal message exchanges. Inf. Sci. **180**(15), 2895–2903 (2010)
3. Cremers, C.: Examining indistinguishability-based security models for key exchange protocols: the case of ck, ck-hmqv, and eck. In: Proceedings of the 6th ACM Symposium on Information, Computer and Communications Security, pp. 80–91. ACM (2011)
4. Fiore, D., Gennaro, R.: Making the Diffie-Hellman protocol identity-based. In: Pieprzyk, J. (ed.) CT-RSA 2010. LNCS, vol. 5985, pp. 165–178. Springer, Heidelberg (2010)
5. Fujioka, A., Suzuki, K., Xagawa, K., Yoneyama, K.: Strongly secure authenticated key exchange from factoring, codes, and lattices. In: Fischlin, M., Buchmann, J., Manulis, M. (eds.) PKC 2012. LNCS, vol. 7293, pp. 467–484. Springer, Heidelberg (2012)
6. Geng, M., Zhang, F.: Provably secure certificateless two-party authenticated key agreement protocol without pairing. In: International Conference on Computational Intelligence and Security, CIS 2009, vol. 2, pp. 208–212. IEEE (2009)
7. Günther, C.G.: An identity-based key-exchange protocol. In: Quisquater, J.-J., Vandewalle, J. (eds.) EUROCRYPT 1989. LNCS, vol. 434, pp. 29–37. Springer, Heidelberg (1990)
8. He, D., Padhye, S., Chen, J.: An efficient certificateless two-party authenticated key agreement protocol. Comput. Math. Appl. **64**(6), 1914–1926 (2012)
9. Islam, S., Biswas, G.: An improved pairing-free identity-based authenticated key agreement protocol based on ecc. Procedia Eng. **30**, 499–507 (2012)
10. Krawczyk, H.: HMQV: a high-performance secure Diffie-Hellman protocol. In: Shoup, V. (ed.) CRYPTO 2005. LNCS, vol. 3621, pp. 546–566. Springer, Heidelberg (2005)
11. LaMacchia, B., Lauter, K., Mityagin, A.: Stronger security of authenticated key exchange. In: Susilo, W., Liu, J.K., Mu, Y. (eds.) ProvSec 2007. LNCS, vol. 4784, pp. 1–16. Springer, Heidelberg (2007)
12. Lippold, G., Boyd, C., Gonzalez Nieto, J.: Strongly secure certificateless key agreement. In: Shacham, H., Waters, B. (eds.) Pairing 2009. LNCS, vol. 5671, pp. 206–230. Springer, Heidelberg (2009)
13. Lippold, G., Nieto, J.G.: Certificateless key agreement in the standard model. In: Proceedings of the Eighth Australasian Conference on Information Security, vol. 105, pp. 75–85. Australian Computer Society Inc. (2010)
14. Saeednia, S.: Improvement of gunther's identity-based key exchange protocol. Electron. Lett. **36**(18), 1535–1536 (2000)
15. Sun, H., Wen, Q., Zhang, H., Jin, Z.: A novel pairing-free certificateless authenticated key agreement protocol with provable security. Front. Comput. Sci. **7**(4), 544–557 (2013)
16. Swanson, C., Jao, D.: A study of two-party certificateless authenticated key-agreement protocols. In: Roy, B., Sendrier, N. (eds.) INDOCRYPT 2009. LNCS, vol. 5922, pp. 57–71. Springer, Heidelberg (2009)

17. Sree Vivek, S., Sharmila Deva Selvi, S., Renganathan Venkatesan, L., Pandu Rangan, C.: Efficient, pairing-free, authenticated identity based key agreement in a single round. In: Susilo, W., Reyhanitabar, R. (eds.) ProvSec 2013. LNCS, vol. 8209, pp. 38–58. Springer, Heidelberg (2013)
18. Yang, G., Tan, C.-H.: Strongly secure certificateless key exchange without pairing. In: Proceedings of the 6th ACM Symposium on Information, Computer and Communications Security, pp. 71–79. ACM (2011)