Efficient ID-based Group Key Agreement with Bilinear Maps

Kyu Young Choi[1], Jung Yeon Hwang[1], and Dong Hoon Lee[2]

Center for Information Security Technologies(CIST),
Korea University, Seoul, Korea
[1] {young,videmot}@cist.korea.ac.kr
[2] {donghlee}@korea.ac.kr

Abstract. In modern collaborative and distributed applications, authenticated group key agreement (GKA) is one of important issues. Recently identity (ID)-based authenticated GKA has been increasingly researched because of the simplicity of a public key management. In this paper, we present a formal treatment on ID-based authenticated GKA, which extends the standard GKA model. We present two GKA protocols which use a bilinear-based cryptography: one is a bilinear variant of Burmester and Desmedt protocol [13] and the other is ID-based authenticated protocol based on the former protocol. Our protocols are scalable and 2-round protocols with forward secrecy. In particular, the ID-based authenticated GKA protocol provides a batch verification technique, which verifies the validity of transcripts from other group players simultaneously and improves computational efficiency. We then prove their securities under the decisional bilinear DH and computational DH assumptions.

1 Intoduction

Background. In many modern collaborative and distributed applications such as multicast communication, audio-video conference and collaborative tools, scalable and reliable group communication is one of the critical problems. A group key agreement (GKA) protocol allows a group of users to share a key which may later be used to achieve some cryptographic goals. In addition to this basic tool an authentication mechanism provides an assurance of key-sharing with intended users. A protocol achieving these two goals is called an authenticated group key agreement (AGKA) protocol.

Among various authentication flavors, asymmetric techniques such as certificate based PKI (public key infrastructure) or ID-based system are commonly used to provide authentication. In a typical PKI deployed system, a user should obtain a certificate of a long-lived public key from the certifying authority and this certificate be given to a partner to authenticate the user. Whereas in a ID-based system, the partner just has to know the public identity of the user such as e-mail address. Thus, compared to certificate-based PKI system, ID-based authenticated systems simplify the key agreement (management) procedures.

Several papers have attempted to establish ID-based authenticated key agreement protocol. But the results in [19, 22, 24] only present informal analysis for the security of the proposed protocols and some of these protocols subsequently found to be flawed [19]. Joux [15] proposed a single round tripartite key agreement using Weil and Tate pairings but unauthenticated. Authenticated versions of this protocol were presented in [1, 24]. Unfortunately, Joux's method does not seem possible to be extended to larger groups consisting of more than three parties since the method is based on the *bilinearity* itself. Recently, an ID-based group (n-party) key agreement protocol which uses the one-way function trees and a pairing is firstly proposed by Reddy, et al. [18] with informal security analysis. Barua, et al. [2] proposed an ID-based multi party key agreement scheme which uses ternary trees. The two protocols above have $\mathcal{O}(\lg n)$ communication rounds.

CONTRIBUTION. In this paper we formally present efficient ID-based authenticated group key agreement, which uses a bilinear-based cryptography. The protocol is a contributory key agreement in which generating a group key is the responsibility not only of the group manager, but also of every group member. Hence it does not impose a heavy computational burden on a particular party, which may cause bottle-neck.

To construct our ID-based AGKA protocol, we first present underlying 2-round GKA protocol, which is a bilinear version of the Burmester and Desmedt (BD) protocol [13]. We should be careful of the conversion since the trivial conversion of the BD protocol into a bilinear setting by simply substituting generators does not provide security even against a passive adversary. This security degradation stems from the *gap* property of a certain group where DDH problem is easy but CDH problem hard.

We then make an ID-based authentication method by combining this method and the former GKA protocol. In fact the presented ID-based authentication method can be naturally transformed into a normal ID-based signature scheme. Moreover the method provides a batch verification technique, which verifies the validity of transcripts simultaneously, to greatly improve computational efficiency. Like the underlying GKA protocol, our ID-based AGKA protocol is 2-round. Our ID-based AGKA protocol is most efficient in computational and communicational costs as compared to other previous known ID-based AGKA protocols.

We prove the security of both protocols under the intractability of CDH and DBDH (Decisional Bilinear DH) problems in the *random oracle* model. The protocols achieve *forward secrecy* in the sense that exposure of user's long-lived secret keys does not compromise the security of previous session keys.

RELATED WORK. Since the original two party Diffie-Hellman key agreement protocol has been presented in [14], authenticated key agreement problems have been extensively researched. In particular, Bellare and Rogaway adapted so-called *provable security* to a key exchange and firstly provided formal framework in two and three party setting [5,6]. Based on that model, many subsequent works have identified concrete cryptographic problems.

Only recently, provably secure solutions for the authenticated group key agreement problem was presented in works of Bresson, et al. [12,10,11], which extended the results in [5,6,4]. Despite of the initial formal step, these protocols, based on the protocols of Steiner, et al. [23], require (relatively) expensive computational cost and the number of round is linear in the number of users participating in a session. Boyd, et al. [8] presented very efficient GKA protocol with a security proof in the random oracle model but did not provide forward secrecy. Katz, et al. [16] presented the constant-round and scalable AGKA protocol with forward secrecy, which is proven secure in the standard model. They took a modular approach and used a signature-based compiler that transforms any GKA protocol secure against a passive adversary to one secure against a stronger active adversary.

Organization. Our paper is organized as follows. We define our security model in Section 2. We review cryptographic assumptions needed in Section 3. We present our GKA and ID-based AGKA protocols and prove the security in Section 4. We finally compare our protocol with other ID-based AGKA protocols and conclude in Section 5.

2 The Model

The model described in this section extends one of Bresson, et al. [10] which follows the approach of Bellare and Rogaway [5,6].

In our protocol, we assume broadcast network in which the users can *broacast* messages to others. Our broadcast network will neither provide authenticity nor guarantee that all user receive identical messages. I.e. we allow the possibility that a malicious adversary may read the broadcast messages and substitute some of them.

2.1 Security Model

Participants. We assume that each user U_i has a unique identity ID_i from $\{0,1\}^\ell$ and all identities are distinct. We also assume for simplicity a fixed set of protocol users $\mathcal{U} = \{U_1, \ldots, U_n\}$ where the number of users is polynomial in the security parameter k.

In the model we allow each user $U_i \in \mathcal{U}$ to execute the protocol many times with different users. *Instances* of a user U_i model distinct, but possibly concurrent executions of the protocol. We denote instance s of a user U_i, called an oracle, by Π_i^s for an integer $s \in \mathbb{N}$.

Initialization. During this phase, each user $U \in \mathcal{U}$ gets public and private keys. ID-based GKA protocol requires the following initialization phase.

- The master secret key msk and global parameters params are generated by algorithm Setup : $\mathsf{params} \leftarrow \mathsf{Setup}(1^k, \ell)$ where ℓ is the identity length.
- Each user U_i gains the long term secret key S_i from algorithm Ext : $S_i \leftarrow \mathsf{Ext}_{\mathsf{msk}}(ID_i)$.

The public parameters params and identities $\mathcal{ID} = \{ID_1, \ldots, ID_n\}$ are known by all users (and also by adversary).

Adversarial model. Normally, the security of a protocol is related to the adversary's ability. The abilities are formally modeled by queries issued by adversaries. We assume that a probabilistic polynomial time (PPT) adversary $\mathcal{A}$ controls the communications completely and can make queries to any instance. The list of queries that $\mathcal{A}$ can make is summarized below:

- $\mathsf{Extract}(ID_U)$: This query allows the adversary to get the long-term private key corresponding to ID_U where $ID_U \notin \mathcal{ID}$.
- $\mathsf{Execute}(\mathcal{ID})$: This query models passive attacks, where the adversary eavesdrops an executions of the protocol. $\mathcal{A}$ gets back the complete transcripts of an honest execution between the users in $\mathcal{ID}$. The number of group members are chosen by the adversary.
- $\mathsf{Send}(\Pi_i^s, M)$: This query allows the adversary to make the user ID_i run the protocol normally. This sends message M to instance Π_i^s which returns the reply generated by this instance.
- $\mathsf{Reveal}(\Pi_i^s)$: This query models the adversary's ability to find session group keys. If an oracle Π_i^s has *accepted*, holding a session group key K, then K is returned to the adversary.
- $\mathsf{Corrupt}(ID_i)$: This query models the attacks revealing the long-term secret key S_i. This does not outputs any internal data of ID_i.
- $\mathsf{Test}(\Pi_i^s)$: This query models the semantic security of a session key. This query is allowed only once by the adversary $\mathcal{A}$. A random bit b is chosen; if $b = 1$ then the session key is returned, otherwise a random value is returned.

In the model we consider two types of adversaries according to their attack types. The attack types are simulated by the queries issued by adversaries. A *passive adversary* is allowed to issue $\mathsf{Execute}$, Reveal, $\mathsf{Corrupt}$, and Test queries, while an *active adversary* is additionally allowed to issue Send and $\mathsf{Extract}$ queries. Even though $\mathsf{Execute}$ query can be simulated using Send queries repeatedly, we use the $\mathsf{Execute}$ query for more exact analysis.

2.2 Security Notions

Session IDS and Partnering. Following [16], we defines session IDS and partnering. The session IDS (SIDS) for an oracle Π_i^s is defined as SIDS(Π_i^s)=$\langle SID_{ij} \rangle$, where SID_{ij} is the concatenation of all messages sent and received by an oracle Π_i^s during the execution. The partner ID for an oracle Π_i^s, denoted by PIDS(Π_i^s), is a set of the identities of the users with whom Π_i^s intends to establish a session key. Instances Π_i^s and Π_j^t are *partnered* if and only if PIDS(Π_i^s)=PIDS(Π_j^t)

and SIDS(Π_i^s)=SIDS(Π_j^t). The presented notion of parting is simple since all messages are sent to all other users taking part in the protocol. We say that an oracle Π_i^s *accepts* when it has enough information to compute a session key.

Freshness. An oracle Π_i^s is said *fresh* (or hold a *fresh* key K) if:
- Π_i^s has accepted a session key $K \neq$ NULL and neither Π_i^s nor one of its partners has been asked for a Reveal query,
- No $\mathsf{Corrupt}$ query has been asked before a query of the form $\mathsf{Send}(\Pi_i^s,*)$ or $\mathsf{Send}(\Pi_j^t,*)$, where Π_j^t is one of Π_i^s's partners.

Definitions of Security. We define the security of the protocol by following game between the adversary $\mathcal{A}$ and an infinite set of oracles Π_i^s for $ID_i \in \mathcal{ID}$ and $s \in \mathbb{N}$.

1. The long-term keys are assigned to each user through the initialization phase related to the security parameter.
2. Run adversary $\mathcal{A}$ who may issue some queries and get back the answers by the corresponding oracles.
3. At some stage during the execution a Test query is issued by the adversary to a *fresh* oracle. The adversary may continue to make other queries, eventually outputs its guess b' for the bit b involved in the Test query and terminates.

In this game, the advantage of the adversary $\mathcal{A}$ is measured by the ability distinguishing the session group key from a random value, i.e. its ability guessing b. We define Succ to be the event that $\mathcal{A}$ correctly guesses the bit b used by the Test oracle in the answering this query. The advantage of an adversary $\mathcal{A}$ in attacking protocol P is defined as $\mathsf{Adv}_{\mathcal{A},P}(k) = \left|2 \cdot Pr[\mathsf{Succ}] - 1\right|$.

We say that a protocol P is a *secure (ID-based authenticated) group key agreement* scheme if the following two properties are satisfied:

– Correctness: in the presence of a (active) passive adversary partner oracles accept the same key.
– Indistinguishability: for every PPT (active) passive adversary $\mathcal{A}$, $\mathsf{Adv}_{\mathcal{A},P}(k)$ is negligible.

Forward Secrecy. In this paper, we are concerned with protocols providing forward secrecy meaning that an adversary gets negligible knowledge information about *previously* established session keys when making a $\mathsf{Corrupt}$ query. We define $\mathsf{Adv}_P^{GKA\text{-}fs}(t, q_{ex})$ to be the maximal advantage of any passive adversary attacking P, running in time t and making q_{ex} $\mathsf{Execute}$ queries. Similarly, we define $\mathsf{Adv}_P^{AGKA\text{-}fs}(t, q_{ex}, q_s)$ to be the maximal advantage of any active adversary attacking P, running in time t and making q_{ex} $\mathsf{Execute}$ queries and q_s Send queries.

Authentication. In this paper, we focus on AGKA with implicit authentication; a key agreement protocol is said to provide implicit key authentication if users are assured that no other users except partners can possibly learn the value of a particular secret key. Note that the property of implicit key authentication does not necessarily mean that partners have actually obtained the key.

3 The Bilinear Maps and Assumptions

In this section, we review some assumptions related to our protocols. Through the paper, we assume that $\mathbb{G}_1$ is a cyclic additive group of prime order q and $\mathbb{G}_2$ is a cyclic multiplicative group of same order q, and the discrete logarithm problem (DLP) in both $\mathbb{G}_1$ and $\mathbb{G}_2$ are intractable.

CDH Parameter Generator: A CDH parameter generator $\mathcal{IG}_{CDH}$ is a PPT algorithm that takes a security parameter 1^k, runs in polynomial time, and outputs an additive group $\mathbb{G}$ of prime order q.

Computational Diffie-Hellman (CDH) problem in $\mathbb{G}$: Informally speaking, the computational DH problem is to compute abP when given a generator P of $\mathbb{G}$ and aP, bP for some $a, b \in \mathbb{Z}_q^*$. More formally, the advantage of $\mathcal{A}$ with respect to $\mathcal{IG}_{BDH}$ is defined to be

$$\Pr\Big[\mathcal{A}(\mathbb{G}, P, aP, bP) = abP \;\Big|\; \mathbb{G} \leftarrow \mathcal{IG}_{CDH}(1^k); P \leftarrow \mathbb{G}; a, b \leftarrow \mathbb{Z}_q^*\Big].$$

$\mathcal{IG}_{CDH}$ is said to satisfy the CDH assumption if any PPT $\mathcal{A}$ has negligible advantage in solving CDH problem.

Admissible Bilinear Map. We call $e : \mathbb{G}_1 \times \mathbb{G}_1 \longrightarrow \mathbb{G}_2$ an *admissible bilinear* map if it saifsfies the following properties:

1. Bilinear : $e(aP, bQ) = e(P, Q)^{ab}$ for all $P, Q \in \mathbb{G}_1$ and $a, b \in \mathbb{Z}_q^*$.
2. Non-degenerate : There exist a $P \in \mathbb{G}_1$ such that $e(P, P) \neq 1$.
3. Computable : There exists an efficient algorithm to compute $e(P, Q)$ for all $P, Q \in \mathbb{G}_1$.

BDH Parameter Generator: A BDH parameter generator $\mathcal{IG}_{BDH}$ is a probabilistic polynomial time (PPT) algorithm that takes a security parameter 1^k, runs in polynomial time, and outputs the description of two groups $\mathbb{G}_1$ and $\mathbb{G}_2$ of the same order q and an admissible bilinear map $e : \mathbb{G}_1 \times \mathbb{G}_1 \longrightarrow \mathbb{G}_2$.

Decisional Bilinear Diffie-Hellman (DBDH) problem in $[\mathbb{G}_1, \mathbb{G}_2, e]$: Informally speaking, the decisional BDH problem is to distinguish between tuples of the form $(P, aP, bP, cP, e(P, P)^{abc})$ and $(P, aP, bP, cP, e(P, P)^d)$ for random $P \in \mathbb{G}_1$, and $a, b, c, d \in \mathbb{Z}_q^*$. More formally, the advantage of $\mathcal{A}$ with respect to $\mathcal{IG}_{BDH}$ is defined to be

$$\left| \Pr\left[\mathcal{A}\left(\mathbb{G}_1, \mathbb{G}_2, e, P, aP, bP, cP, e(P,P)^{abc}\right) = 1 \;\middle|\; \begin{array}{c}(\mathbb{G}_1, \mathbb{G}_2, e) \leftarrow \mathcal{IG}_{BDH}(1^k);\\ P \leftarrow \mathbb{G}_1; a, b, c \leftarrow \mathbb{Z}_q^*\end{array}\right] \right.$$
$$\left. -\Pr\left[\mathcal{A}\left(\mathbb{G}_1, \mathbb{G}_2, e, P, aP, bP, cP, e(P,P)^{d}\right) = 1 \;\middle|\; \begin{array}{c}(\mathbb{G}_1, \mathbb{G}_2, e) \leftarrow \mathcal{IG}_{BDH}(1^k);\\ P \leftarrow \mathbb{G}_1; a, b, c, d \leftarrow \mathbb{Z}_q^*\end{array}\right] \right|.$$

$\mathcal{IG}_{BDH}$ is said to satisfy the DBDH assumption if any PPT $\mathcal{A}$ has negligible advantage in solving DBDH problem.

As noted in [7], BDH parameter generators satisfying the DBDH assumption is believed to be constructed from Weil and Tate pairings associated with supersingular eliptic curves or Abelian varieties.

4 Our GKA and ID-based AGKA protocol

4.1 GKA Protocol Using a Bilinear Map

We now describe a 2-round GKA protocol using bilinear maps. We denote this protocol by B-GKA. In fact, this protocol is a bilinear variant of the protocol by Burmester and Desmedt. In this protocol, no long-term public/private keys are required. In the following description groups $\mathbb{G}_1$, $\mathbb{G}_2$ and a bilinear map e are generated by a BDH generator in Section 3 and P is a random generator of $\mathbb{G}_1$. When n users U_1,..,U_n want to establish a session key, they proceed as follows :

[**Round 1**] Each user U_i picks a random integer $a_i \in \mathbb{Z}_q^*$ and computes $P_i = a_i P$. Then each U_i broadcasts P_i to all others and keeps a_i secret.

[**Round 2**] Upon receipt of P_{i-1}, P_{i+1} and P_{i+2}, each users U_i computes

$$D_i = e\big(a_i(P_{i+2} - P_{i-1}),\ P_{i+1}\big)$$

and broadcasts D_i to all others.

Key Computation. Each U_i computes the session key as follows :

$$K_i = e(a_i P_{i-1}, P_{i+1})^n D_i^{n-1} D_{i+1}^{n-2} \cdots D_{i-2}.$$

It is obvious that all honest users compute the same key as follows:

$$K = e(P,P)^{a_1 a_2 a_3 + \cdots + a_{n-1} a_n a_1 + a_n a_1 a_2}.$$

We note that the trivial conversion of the BD protocol to a bilinear setting by simply substituting generators does not provide security even against a passive adversary. This is possible because of the gap property of $\mathbb{G}_1$ where DDH problem is easy but CDH problem hard.

Theorem 1. *The protocol* B-GKA *is a secure GKA protocol providing forward secrecy under the DBDH assumption. Concretely,*

$$\mathrm{Adv}_{\mathsf{B\text{-}GKA}}^{\mathrm{GKA-fs}}(t, q_{ex}) \le 4 \cdot \mathrm{Adv}_{\mathbb{G}_1,\mathbb{G}_2,\mathrm{e}}^{\mathrm{DBDH}}(t).$$

We can prove Theorem 1 in two steps by using standard hybrid argument and showing information theoretical independence of a secret key. The security analysis is similar to that of Katz, et al.[17]. For space limitation we omit the proof. However a tighter security reduction can be obtained using *random self-reducibility* properties of the DBDH problem. The method of the reduction in [17, 21] is similarly applied to our reduction.

4.2 ID-based Authenticated Group Key Agreement Protocol

In this section we present an ID-based AGKA protocol based on the previous protocol B-GKA. We denote this protocol by ID-GKA. The protocol involves the trusted key generation center (KGC). In the following description $H : \{0,1\}^* \longrightarrow \mathbb{Z}_q$ and $H_1 : \{0,1\}^* \longrightarrow \mathbb{G}_1$ are cryptographic hash functions. H and H_1 are considered as random oracles in the security proof.

Setup. KGC runs BDH parameter generator, and chooses a random $s \in \mathbb{Z}_q^*$ and a generator P of $\mathbb{G}_1$ and computes $P_{pub} = sP$. Then KGC keeps s secret as the *master secret key* and publishes system parameters params $= \{e, \mathbb{G}_1, \mathbb{G}_2, q, P, P_{pub}, H, H_1\}$.

Extract. When a user with identity ID wishes to obtain a key pair, KGC computes $Q_{ID} = H_1(ID)$ and the private key $S_{ID} = sQ_{ID}$, and returns S_{ID} to the user.

Let $\{U_1, ..., U_n\}$ be a set of users who want to establish a session key and ID_i be the identity of each U_i. The indices are subject to modulo n. U_i's long-term public and private key pair is $\langle ID_i,\ S_i = sQ_i\rangle$.

[**Round 1**] Each user U_i picks a random integer $a_i \in \mathbb{Z}_q^*$ and computes $P_i = a_iP$, $h_i = H(P_i)$ and $T_i = a_iP_{pub} + h_iS_i$. Each U_i broadcasts $\langle P_i, T_i\rangle$ to all others and keeps a_i secret.

[**Round 2**] Upon the receipt of $\langle P_{i-1}, T_{i-1}\rangle$, $\langle P_{i+1}, T_{i+1}\rangle$ and $\langle P_{i+2}, T_{i+2}\rangle$, each user U_i checks if the following equation holds:

$$e\Big(\sum_{k\in\{-1,1,2\}} T_{i+k}, P\Big) = e\Big(\sum_{k\in\{-1,1,2\}} (P_{i+k} + h_{i+k}Q_{i+k}), P_{pub}\Big)$$

If the above equation is satisfied, then U_i computes $D_i = e\big(a_i(P_{i+2} - P_{i-1}),\ P_{i+1}\big)$ and broadcasts D_i to all others. Otherwise U_i stops.

Key Computation. Each U_i computes the session key,

$$K_i = e(a_iP_{i-1}, P_{i+1})^n D_i^{n-1} D_{i+1}^{n-2} \cdots D_{i-2}.$$

The correctness of key computation is same to that of the protocol B-GKA.
In the above protocol, we used an authentication scheme Γ defined as follows;

Generation. Given a secret key $S_{ID}{=}sH_1(ID)$, compute $T = aP_{pub} + hS_{ID}$ where $a \in_R \mathbb{Z}_q^*$ and $h = H(aP)$; $\langle aP, T\rangle \leftarrow \Gamma_{gen}(S_{ID})$.

Verification. Given a public Q_{ID} and $\langle aP, T\rangle$, verify that $e(T, P) = e(aP + hQ_{ID}, P_{pub})$, where $h = H(aP)$; `True` or `False` $\leftarrow \Gamma_{ver}(Q_{ID}, \langle aP, T\rangle)$.

The correctness of Γ is easily proved as follows; for given public Q_{ID} and $\langle aP, T\rangle$, $e(T, P) = e(asP + hsQ_{ID}, P) = e(aP + hQ_{ID}, P_{pub})$ where $h = H(aP)$.

In fact, in Round 2, each user uses a *screening* test [3] to verify the validity of authentication for computational efficiency. This test provides a weaker notion determining if each user has at some point generated the transcript for authentication rather than checking the given data is a valid transcript for authentication. This validation notion is adequate for our goal since each user wants to do implicit authentication for a session rather than to have an authentication data. However we can directly adapt a batch technique providing a strong notion, such like random subset test and small exponent test, etc., as in [3, 9].

We note that the authentication scheme Γ can be easily transformed into an ID-based signature scheme.

For the following security analysis we define $Forger_\Gamma$ as a PPT forger of the authentication scheme Γ under the adaptively chosen ID attack and $Forger_\Gamma^{ID}$ a PPT forger of Γ under given ID attack.

Theorem 2. *Suppose the hash functions H, H_1 are random oracles. Then the protocol* ID-GKA *is a secure AGKA protocol providing forward secrecy under the DBDH assumption and the CDH assumption. Concretely,*

$$\mathrm{Adv}_{\mathsf{ID\text{-}GKA}}^{\mathrm{AGKA-fs}}(t, q_{ex}, q_s) \leq 2nq_{ex} \cdot \mathrm{Adv}_{\mathbb{G}_1,\mathbb{G}_2,\mathrm{e}}^{\mathrm{DBDH}}(t) + \mathrm{Adv}_{\Gamma}^{\mathrm{Forge}}(t).$$

where $\mathrm{Adv}_{\Gamma}^{\mathrm{Forge}}(t)$ *is the maximum advantage of any $Forger_\Gamma$ running in time t.*

Proof. Let $\mathcal{A}$ be an active adversary that gets advantage in attacking ID-GKA. The adversary $\mathcal{A}$ can get an advantage by forging authentication transcripts, namely impersonating a user or 'breaking' the protocol without altering transcripts.

First we assume that $\mathcal{A}$ breaks ID-GKA by using adaptive impersonation ability. Using $\mathcal{A}$, we can construct a $Forger_\Gamma$ $\mathcal{C}$ that generates a valid message pair $\langle ID, aP, T\rangle$ with respect to Γ as follows: a $Forger_\Gamma$ $\mathcal{C}$ honestly generates all other public and private keys for the system. $\mathcal{C}$ simulates the oracle queries of $\mathcal{A}$ in the natural way; this results in a perfect simulation unless $\mathcal{A}$ queries Corrupt(ID). If this occurs, $\mathcal{C}$ simply aborts. Otherwise, if $\mathcal{A}$ generates a new and valid message pair $\langle ID, aP, T\rangle$, this event is denoted by Forge, then $\mathcal{C}$ generates the message pair $\langle ID, aP, T\rangle$. The success probability of $\mathcal{C}$ satisfies $\Pr_{\mathcal{A}}[\mathsf{Forge}] \leq \mathrm{Adv}_{\mathcal{C},\Gamma}^{\mathrm{Forge}}(t) \leq \mathrm{Adv}_{\Gamma}^{\mathrm{Forge}}(t)$.

Next we assume that $\mathcal{A}$ breaks ID-GKA without altering transcripts. Before describing the details we define the Modified DBDH(MDBDH) problem related to our security reduction. The MDBDH problem in $[\mathbb{G}_1, \mathbb{G}_2, e]$ is to distinguish between tuples of the form $(P, aP, bP, cP, sP, saP, sbP, scP, e(P,P)^{abc})$ and $(P, aP, bP, cP, sP, saP, sbP, scP, e(P,P)^{d})$ for random $P \in \mathbb{G}_1$, and $a, b, c, d, s \in \mathbb{Z}_q^*$. It is easily showed that the DBDH problem and the MDBDH problem in $[\mathbb{G}_1, \mathbb{G}_2, e]$ are computationally equivalent. Namely, $\mathrm{Adv}_{\mathbb{G}_1,\mathbb{G}_2,\mathrm{e}}^{\mathrm{DBDH}}(t) = \mathrm{Adv}_{\mathbb{G}_1,\mathbb{G}_2,\mathrm{e}}^{\mathrm{MDBDH}}(t)$.

We first consider the case that an adversary $\mathcal{A}$ makes only a single Execute query Execute($ID_1, ..., ID_n$) and then extend this to the case that $\mathcal{A}$ makes multiple Execute queries. Let n be the number of users chosen by the adversary $\mathcal{A}$. The distribution of the transcript $\mathcal{T}$ and the resulting group key K is given by:

$$\mathsf{params} = \left[\begin{array}{c} (\mathbb{G}_1, \mathbb{G}_2, e) \leftarrow \mathcal{IG}_{BDH}(1^k);\ P \leftarrow \mathbb{G}_1;\ s \leftarrow \mathbb{Z}_q^*;\ P_{pub} = sP \\ Q_1, ..., Q_n \leftarrow \mathbb{G}_1;\ S_1 = sQ_1, ..., S_n = sQ_n : (\mathbb{G}_1, \mathbb{G}_2, e, P, P_{pub}) \end{array}\right]$$

$$Real = \left[\begin{array}{c} a_1, ..., a_n, h_1, ..., h_n \leftarrow \mathbb{Z}_q^*;\ P_1 = a_1 P, .., P_n = a_n P; \\ T_1 = a_1 P_{pub} + h_1 S_1, ..., T_n = a_n P_{pub} + h_n S_n; \\ D_1 = \frac{e(a_1 P_2, P_3)}{e(a_1 P_n, P_2)}, D_2 = \frac{e(a_2 P_3, P_4)}{e(a_2 P_1, P_3)}, ..., D_n = \frac{e(a_n P_1, P_2)}{e(a_n P_{n-1}, P_1)}; \\ \mathcal{T} = \langle P_1, ..., P_n, T_1, ..., T_n, D_1, ..., D_n\rangle; \\ K = e(a_1 P_n, P_2)^n D_1^{n-1} \cdots D_{n-1} : (\mathcal{T}, K) \end{array}\right]$$

Consider the distributions $Fake_i$ defined as follows:

$$Fake_1 = \left[\begin{array}{c} r_{n,1,2}, a_1, ..., a_n, h_1, ..., h_n \leftarrow \mathbb{Z}_q^*;\ P_1 = a_1P, .., P_n = a_nP; \\ T_1 = a_1P_{pub} + h_1S_1, ..., T_n = a_nP_{pub} + h_nS_n; \\ D_1 = \frac{e(a_1P_2,P_3)}{e(r_{n,1,2}P,P)}, D_2 = \frac{e(a_2P_3,P_4)}{e(a_2P_1,P_3)}, ..., D_n = \frac{e(r_{n,1,2}P,P)}{e(a_nP_{n-1},P_1)}; \\ \mathcal{T} = \langle P_1, ..., P_n, T_1, ..., T_n, D_1, ..., D_n \rangle; \\ K = e(r_{n,1,2}P,P)^n D_1^{n-1} \cdots D_{n-1} : (\mathcal{T}, K) \end{array}\right] \cdots$$

Continuing in this way, we obtain the distribution:

$$Fake_n = \left[\begin{array}{c} r_{n,1,2}, ..., r_{n-1,n,1}, a_1, ..., a_n, h_1, ..., h_n \leftarrow \mathbb{Z}_q^*;\ P_1 = a_1P, .., P_n = a_nP; \\ T_1 = a_1P_{pub} + h_1S_1, ..., T_n = a_nP_{pub} + h_nS_n; \\ D_1 = \frac{e(r_{1,2,3}P,P)}{e(r_{n,1,2}P,P)}, D_2 = \frac{e(r_{2,3,4}P,P)}{e(r_{1,2,3}P,P)}, ..., D_n = \frac{e(r_{n,1,2}P,P)}{e(r_{n-1,n,1}P,P)}; \\ \mathcal{T} = \langle P_1, ..., P_n, T_1, ..., T_n, D_1, ..., D_n \rangle; \\ K = e(r_{n,1,2}P,P)^n D_1^{n-1} \cdots D_{n-1} : (\mathcal{T}, K) \end{array}\right]$$

$\mathcal{A}$ can compute all $a_iP_{pub} = T_i - hS_i$ from the transcripts since $\mathcal{A}$ can obtain all secret keys S_i and hash values h_i $(i = 1, ..., n)$ by using multiple Corrupt and H queries, respectively. Therefore the distribution of previous transcripts is changed by the distribution related to the modified DBDH problem. Let $\varepsilon(t) = \mathrm{Adv}^{\mathrm{MDBDH}}_{\mathbb{G}_1,\mathbb{G}_2,\mathrm{e}}(t)$. A standard argument shows that for any algorithm $\mathcal{A}$ running in time t we have:

$$\left|Pr[\mathcal{T}\leftarrow Real; K\leftarrow Real\colon \mathcal{A}(\mathcal{T},K){=}1] - Pr[\mathcal{T}\leftarrow Fake_1; K\leftarrow Fake_1\colon \mathcal{A}(\mathcal{T},K){=}1]\right| \le \varepsilon(t)$$
$$\left|Pr[\mathcal{T}\leftarrow Fake_1; K\leftarrow Fake_1\colon \mathcal{A}(\mathcal{T},K){=}1] - Pr[\mathcal{T}\leftarrow Fake_2; K\leftarrow Fake_2\colon \mathcal{A}(\mathcal{T},K){=}1]\right| \le \varepsilon(t)$$
$$\vdots$$
$$\left|Pr[\mathcal{T}\leftarrow Fake_{n-1}; K\leftarrow Fake_{n-1}\colon \mathcal{A}(\mathcal{T},K){=}1] - Pr[\mathcal{T}\leftarrow Fake_n; K\leftarrow Fake_n\colon \mathcal{A}(\mathcal{T},K){=}1]\right| \le \varepsilon(t).$$

Let $e(P,P){=}g$ in $\mathbb{G}_2$. In experiment $\mathrm{Fake_n}$, the values $r_{1,2,3}, ..., r_{n,1,2}$ are constrained by $\mathcal{T}$ according to the following n equations $\log_g D_1 = r_{1,2,3} - r_{n,1,2}$, $\log_g D_2 = r_{2,3,4} - r_{1,2,3}$,...,$\log_g D_n = r_{n,1,2} - r_{n-1,n,1}$ of which only $n-1$ of these are linearly independent. Furthermore, K may be expressed as $K = e(P,P)^{a_na_1a_2+\cdots+a_{n-1}a_na_1}$; equivalently, we have

$$\log_g K = r_{1,2,3} + r_{2,3,4} + \cdots + r_{n,1,2}.$$

Since this final equation is linearly independent from the set of equations above, the value of K is independent of $\mathcal{T}$. This implies that, for any adversary $\mathcal{A}$:

$$\left|Pr[\mathcal{T}\leftarrow Fake_n; K\leftarrow Fake_n\colon \mathcal{A}(\mathcal{T},K){=}1] = Pr[\mathcal{T}\leftarrow Fake_n; K\leftarrow Random\colon \mathcal{A}(\mathcal{T},K){=}1]\right|.$$

Similarly, a standard argument shows that for any algorithm $\mathcal{A}$ running in time t we have:

$$\left|Pr[\mathcal{T}\leftarrow Fake_n; K\leftarrow Fake_n\colon \mathcal{A}(\mathcal{T},K){=}1] - Pr[\mathcal{T}\leftarrow Fake_{n-1}; K\leftarrow Random\colon \mathcal{A}(\mathcal{T},K){=}1]\right| \le \varepsilon(t)$$
$$\vdots$$
$$\left|Pr[\mathcal{T}\leftarrow Fake_1; K\leftarrow Random\colon \mathcal{A}(\mathcal{T},K){=}1] - Pr[\mathcal{T}\leftarrow Real; K\leftarrow Random\colon \mathcal{A}(\mathcal{T},K){=}1]\right| \le \varepsilon(t)$$

Eventually, we obtain the equation as follow:

$$\left|Pr[\mathcal{T}\leftarrow Real; K\leftarrow Real\colon \mathcal{A}(\mathcal{T},K){=}1] - Pr[\mathcal{T}\leftarrow Real; K\leftarrow Random\colon \mathcal{A}(\mathcal{T},K){=}1]\right| \le 2n\varepsilon(t)$$

Since $\epsilon(t) = \mathrm{Adv}^{\mathrm{MDBDH}}_{\mathbb{G}_1,\mathbb{G}_2,\mathrm{e}}(t) = \mathrm{Adv}^{\mathrm{DBDH}}_{\mathbb{G}_1,\mathbb{G}_2,\mathrm{e}}(t)$, we have the result that the advantage of $\mathcal{A}$ conditioned on the event $\sim$Forge is bounded by $2n \cdot \mathrm{Adv}^{\mathrm{DBDH}}_{\mathbb{G}_1,\mathbb{G}_2,\mathrm{e}}(t)$. Hence we have

$$\mathrm{Adv}_{\textit{ID-GKA}}^{\mathrm{AGKA-fs}}(t, 1, q_s) \leq 2n \cdot \mathrm{Adv}_{\mathbb{G}_1,\mathbb{G}_2,\mathrm{e}}^{\mathrm{DBDH}}(t) + \mathrm{Adv}_{\Gamma}^{\mathrm{Forge}}(t).$$

Finally we have the desired result by adapting a standard hybrid argument that

$$\mathrm{Adv}_{\textit{ID-GKA}}^{\mathrm{AGKA-fs}}(t, q_{ex}, q_s) \leq 2nq_{ex} \cdot \mathrm{Adv}_{\mathbb{G}_1,\mathbb{G}_2,\mathrm{e}}^{\mathrm{DBDH}}(t) + \mathrm{Adv}_{\Gamma}^{\mathrm{Forge}}(t).$$

We next show that the authentication scheme Γ is secure against existential forgery on adaptively chosen ID attack.

Lemma 1. *Let the hash function H_1 be random oracle. Suppose there exists a $Forger_\Gamma$ $\mathcal{A}$ for an adaptively chosen ID with running time t_0 and advantage ε_0. Suppose $\mathcal{A}$ makes at most q_{H_1} queries to the hash function H_1. Then a $Forger_\Gamma^{ID}$ $\mathcal{B}$ for a given ID with running time $t_1 \leq t_0$ has advantage $\varepsilon_1 \leq \varepsilon_0(1 - \frac{1}{q})/q_{H_1}$.*

Lemma 2. *Let the hash function H, H_1 be random oracles. Suppose there exists a $Forger_\Gamma^{ID}$ $\mathcal{A}$ for a given ID with running time t_1 and advantage $\varepsilon_1 \geq 10(q_s + 1)(q_s + q_H)/q$. Suppose $\mathcal{A}$ makes at most q_H, q_{H_1}, q_s and q_{ex} queries to the H, H_1, Send and Extract respectively. Then there exists an attacker $\mathcal{B}$ that can solve the CDH problem within expected time $t_2 \leq 120686 q_H t_1/\varepsilon_1$.*

The proofs of the above two lemmas are given in Appendix. Combining the Lemma 1 and 2, we obtain that $\mathrm{Adv}_{\Gamma}^{\mathrm{Forge}}(t)$ is negligible in the following theorem. Therefore we can show that our ID-GKA is a secure AGKA providing forward secrecy.

Theorem 3. *Let the hash functions H, H_1 be random oracles. Suppose there exists a $Forger_\Gamma$ $\mathcal{A}$ for an adaptively chosen ID with running time t_0 and advantage $\varepsilon_0 \geq 10q_{H_1}(q_s + 1)(q_s + q_H)/(q - 1)$. Suppose $\mathcal{A}$ makes at most q_H, q_{H_1}, q_s and q_{ex} queries to the H, H_1,Send and Extract respectively. Then there exists an attacker $\mathcal{B}$ that can solve the CDH problem within expected time $t_2 \leq 120686 q_H t_0/\varepsilon_0$.*

5 Comparison and Conclusion

We now compare our protocol ID-GKA with other previously known ID-based GKA protocols, the binary tree based 2T-IDAGKA [18] and the ternary tree based 3T-IDAGKA [2] in Table 1. We use notations as follows:

- *Round*: The total number of rounds.
- *Message*: The total number of messages sent by users.
- *Computation*: The total number of scalar multiplications.
- *Pairing*: The total number of pairing-computations.

Because the number of users is relatively small in practice, we can assume that, in our ID-GKA, the key computation step requires just one scalar multiplication.

Protocol	Round	Message	Computation	Pairing
2T-IDAGKA [18]	$\mathcal{O}(\lg n)$	$\mathcal{O}(n \lg n)$	$\mathcal{O}(n \lg n)$	$\mathcal{O}(n \lg n)$
3T-IDAGKA [2]	$\mathcal{O}(\lg n)$	$\mathcal{O}(n)$	$\mathcal{O}(n)$	$\mathcal{O}(n \lg n)$
Our ID-GKA protocol	$\mathcal{O}(1)$	$\mathcal{O}(n)$	$\mathcal{O}(n)$	$\mathcal{O}(n)$

Table 1. Comparison of ID-based AGKA protocols

As we shown in Table 1, our protocol is the most efficient one as compared to other protocols. In particular, our protocol require $\mathcal{O}(1)$ round and only $\mathcal{O}(n)$ pairing-computations.

In this paper, we have presented a 2-round and scalable ID-based AGKA protocol based on a bilinear variant of the BD protocol [13]. Moreover, we have adapted batch verification technique verifying the validity of transcripts simultaneously, which greatly improves the computational efficiency. We have proved the security of both protocols under the intractability of CDH and DBDH.

References

1. S. Al-Riyami and K.G. Paterson, *Tripartite Authenticated Key Agreement Protocols from Pairings*, Cryptology ePrint Archive, Report 2002/035, 2002. http://eprint.iacr.org/.
2. R. Barua, R. Dutta and P. Sarker, *Extending Joux's Protocol to Multi Party Key Agreement.*, Proc. of Indocrypt '03.
3. M. Bellare, J. A. Garay and T. Rabin, *Fast Batch Verification for modular Exponentiation and Digital Signatures,*Proc. of Eurocrypt '98, LNCS 1403, pp.236-250, Springer-Verlag, 1998.
4. M. Bellare, D. Pointcheaval, and P. Rogaway, *Authenticated key exchange secure against dictionary attacks.*, Proc. of Eurocrypt '00, LNCS 1807, pp.139-155, Springer-Verlag, 2000.
5. M. Bellare, P. Rogaway, *Entity authentication and key distribution*, Proc. of Crypto '93, pp.232-249.
6. M. Bellare, P. Rogaway, *Provably-Secure Session Key Distribution : The Three Party Case*, Proc. of STOC '95, pp. 57-66.
7. D. Boneh and M. Franklin, *Identity-based encryption from the Weil pairing*, Proc. of Crypto '01, LNCS 2139, pp.213-229, Springer-Verlag, 2001.
8. C. Boyd and J.M.G. Nieto, *Round-Optimal Contributory Conference Key Agreement*, Proc. of PKC '03, LNCS 2567, pp.161-174. Springer-Verlag, 2003.
9. C. Boyd and C. Pavlovski, *Attacking and Repairing Batch Verification Schemes*, Proc. of Asiacrypt '00, LNCS 1976, pp.58-71, Springer-Verlag, 2000.
10. E. Bresson, O. Chevassut and D. Pointcheval, *Provably Authenticated Group Diffie-Hellman Key Exchange-The Dynamic Case*, Proc. of Asiacrypt '02, LNCS 2248, pp.290-309, Springer-Verlag, 2001.
11. E. Bresson, O. Chevassut and D. Pointcheval, *Dynamic Group Diffie-Hellman Key Exchange under Standard Assumption (Full version)*, Proc. of Eurocrypt '02, LNCS 2332, pp.321-336. Springer-Verlag, 2002.
12. E. Bresson, O. Chevassut, D. Pointcheval, J.-J. Quisquater, *Provably Authenticated Group Diffie-Hellman Key Exchange*, In Proc. of 8th ACM CCCS, pp.255-264, 2001.
13. M. Burmester and Y. Desmedt, *A Secure and Efficient Conference Key Distribution System*, Proc. of Eurocrypt '94, pp.267-275, LNCS 950, Springer-Verlag, 1994.
14. W. Diffie and M. Hellman, *New Directions in Cryptography*, IEEE Transactions on Information Theory 22(6), pp.644-654, 1976.
15. Joux, *A one round protocol for tripartite Diffie-Hellman*, In W. Bosma, editor, Proceedings of Algorithmic Number Theory Symposium. ANTS IV, LNCS 1838, pp.385-394, Springer-Verlag, 2000.

16. J. Katz and M. Yung, *Scalable Protocols for Authenticated Group Key Exchange*, Proc. of Crypto 2003, LNCS 2729, pp.110-125, Springer-Verlag, 2003.
17. J. Katz and M. Yung, *Scalable Protocols for Authenticated Group Key Exchange*, full version.
18. D.Nalla and K. C. Reddy, *Identity Based Authenticated Group Key Agreement Protocol*, Proc. of Indocrypt '02, LNCS 2551, pp.215-233, Springer-Verlag, 2002.
19. D. Nalla, K. C. Reddy, *ID-based tripartite Authenticated Key Agreement Protocols from pairings*, Cryptology ePrint Archive, Report 2003/004. http://eprint.iacr.org/2003/004/.
20. D. Pointcheval and J. Stern, *Security arguments for digital signatures and blind signatures*, J. of Cryptology, Vol. 13, pp.361-396, 2000.
21. V. Shoup, *On formal models for secure key exchange.* , In ACM Security '99.
22. N.P.Smart, *An Identity based authenticated Key Agreement protocol based on the Weil pairing*, Cryptology ePrint Archive, Report 2001/111,2001. http://eprint.iacr.org/.
23. M. Steiner, G. Tsudik and M. Waidner, *Key Agreement in Dynamic Peer Groups*, IEEE Trans. on Parallel and Distributed Systems 11(8), pp.769-780, 2000.
24. F. Zhang, S. Liu and K. Kim, *ID-based One Round Authenticated Tripartite Key Agreement Protocols with Pairings*, Cryptology ePrint Archive 2002. http://eprint.iacr.org/.

A Proof of Lemma 1

$\mathcal{B}$ is given ID^*. Without any loss of generality, we assume that for any ID, $\mathcal{A}$ makes H_1, Send and Extract queries at most once, and Send and Extract queries for public key are preceded by H_1 hash query.

To respond to these queries $\mathcal{B}$ maintains a list L_{H_1} of $\langle ID_i, Q_i \rangle$. The list is initially empty. First, $\mathcal{B}$ chooses $\alpha \in \{1, ..., q_{H_1}\}$ randomly. $\mathcal{B}$ interacts with $\mathcal{A}$ as follows:

- When $\mathcal{A}$ makes the α-th H_1 query on ID, $\mathcal{B}$ issues a H_1 query for ID^* and returns the result Q^* to $\mathcal{A}$. Then $\mathcal{B}$ adds $\langle ID, Q^* \rangle$ to L_{H_1}. Otherwise, $\mathcal{B}$ issues a H_1 query for ID and returns the result to $\mathcal{A}$. Then $\mathcal{B}$ inserts $\langle ID, Q \rangle$ into L_{H_1}.
- When $\mathcal{A}$ issues an Extract query on Q_i, if $Q_i = Q^*$, then $\mathcal{B}$ outputs FAIL and aborts. Otherwise, $\mathcal{B}$ issues an Extract query for Q_i and returns the result S_i to $\mathcal{A}$.
- When $\mathcal{A}$ issues a H query on a_iP, $\mathcal{B}$ issues a H query for aP and returns the result $H(a_iP)$ to $\mathcal{A}$.
- When $\mathcal{A}$ issues a Send query on ID_i, $\mathcal{B}$ issues a Send query for ID_i and returns the result $\langle ID_i, a_iP, T_i \rangle$ to $\mathcal{A}$.

Eventually, $\mathcal{A}$ outputs $\langle ID', a'P, T' \rangle$. Then $\mathcal{B}$ finds the tuple of the form $\langle ID', Q' \rangle$ in L_{H_1}. If $Q' = Q^*$ then $\mathcal{B}$ outputs $\langle ID^*, a'P, T' \rangle$. Otherwise, $\mathcal{B}$ outputs FAIL and aborts.

To complete the proof of Lemma 1 it remains to calculate the probability that algorithm $\mathcal{B}$ aborts during the simulation. Notice that, if $Q' \neq Q^*$ and a pair $\langle ID', Q' \rangle$ is not found in L_{H_1}, then the output $\langle ID', a'P, T' \rangle$ is independent of the knowledge $\mathcal{A}$ accumulated from its various queries. This means that $\mathcal{A}$ succeeds in this case with probability $1/q$. Therefore, the probability that $\mathcal{B}$ does not abort during the simulation is $\frac{1}{q_{H_1}}(1-\frac{1}{q})$.

B Proof of Lemma 2

First, a BDH parameter generator is run and $\langle e, \mathbb{G}_1, \mathbb{G}_2 \rangle$ is outputted. Then $\mathcal{B}$ receives a CDH instance (P, xP, yP) for randomly chosen $x, y \in \mathbb{Z}_q^*$ and $P \in \mathbb{G}_1$. Its goal is to compute xyP.

$\mathcal{B}$ runs a $Forger_T^{ID}$ $\mathcal{A}$ as a subroutine and simulates its attack environment. $\mathcal{B}$ sets the public system parameters params=$\langle e, \mathbb{G}_1, \mathbb{G}_2, P, P_{pub}, ID^*, H, H_1 \rangle$ by letting $P_{pub} = xP$ where x is the master secret key, which is unknown to $\mathcal{B}$ and selecting an identity ID^* for given ID attack of $\mathcal{A}$. $\mathcal{B}$ gives params to $\mathcal{A}$. Note that, for given ID, the corresponding private key associated to params is $S_{ID} = xQ_{ID} = xH_1(ID)$.

Without loss of generality, we assume that for any ID, $\mathcal{A}$ queries H_1, Send and Extract at most once, and Send and Extract queries for public keys are preceded by an H_1 hash query. To avoid collision and consistently respond to these queries $\mathcal{B}$ maintains two lists L_{H_1} and L_T of $\langle ID_i, r_i, Q_i \rangle$ and $\langle ID_j, a_jP \rangle$, respectively. The lists are initially empty. Algorithm $\mathcal{B}$ interacts with $\mathcal{A}$ as follows:

- When $\mathcal{A}$ issues $H_1(ID^*)$ query, $\mathcal{B}$ returns $Q^* = yP$. For all other H_1 queries, $\mathcal{B}$ picks a random $r_i \in \mathbb{Z}_q^*$ and adds $\langle ID_i, r_i, Q_i \rangle$ to L_{H_1}, and returns $Q_i = r_iP$ to $\mathcal{A}$.
- When $\mathcal{A}$ issues Extract query on Q_i, if $Q_i = Q^*$, then $\mathcal{B}$ outputs FAIL and aborts. Otherwise, $\mathcal{B}$ finds the tuple of the form $\langle ID_i, r_i, Q_i \rangle$ in L_{H_1}, and returns private keys $r_iP_{pub} = r_ixP = xr_iP = xQ_i$ to $\mathcal{A}$.
- When $\mathcal{A}$ issues an H query for a_iP, then $\mathcal{B}$ picks a random $h_i \in \mathbb{Z}_q^*$ and returns h_i to $\mathcal{A}$.
- When $\mathcal{A}$ issues a Send query on ID_i, $\mathcal{B}$ picks a random $a_i \in \mathbb{Z}_q^*$ and computes a_iP, and adds the tuple $\langle ID_i, a_iP \rangle$ to L_T. $\mathcal{B}$ finds the tuple of the form $\langle ID_i, r_i, Q_i \rangle$ in L_{H_1}. Then $\mathcal{B}$ computes $T_i = a_ixP + h_ir_ixP$=$a_iP_{pub} + h_iS_i$ and returns $\langle ID_i, a_iP, T_i \rangle$ to $\mathcal{A}$.

Eventually, $\mathcal{A}$ outputs a valid tuple$\langle ID^*, aP, h, T \rangle$ such that $\langle ID^*, aP \rangle \notin L_T$, which is expected to be valid for the fixed ID^*, without accessing any oracles expect H. By replays of $\mathcal{B}$ with the same random tape but different choices of H, as done in the *forking lemma*(theorem 3)[20], $\mathcal{A}$ outputs two valid tuples $\langle ID^*, aP, h, T \rangle$ and $\langle ID^*, aP, h', T' \rangle$ such that $h \neq h'$. If both outputs are expected ones, then $\mathcal{B}$ computes $(T-T')/(h-h') = xyP$ and outputs it. Otherwise, $\mathcal{B}$ outputs FAIL and aborts.

The total running time t_2 of $\mathcal{B}$ is equal to the running time of the *forking lemma*(theorem 3)[20] which is bounded by $120686q_Ht_1/\varepsilon_1$, as desired.